AD_____

Award Number: DAMD17-98-2-8002

TITLE: Disaster Relief and Emergency Medical Services Project (DREAMS TM): Digital EMS

PRINCIPAL INVESTIGATORS: James H. Duke, Jr., M.D., James Wall, Ph.D., Jose Salinas, Ph.D., R. Douglas Tindall

CONTRACTING ORGANIZATION: University of Texas Health Sciences Center at Houston
Houston, Texas 77225-0708

REPORT DATE: October 2000

TYPE OF REPORT: Final

PREPARED FOR: U.S. Army Medical Research and Materiel Command
Fort Detrick, Maryland 21702-5012

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	October 2000	Annual (23 Jan 99 - 22 Sep 00)

4. TITLE AND SUBTITLE
Disaster Relief and Emergency Medical Services Project (DREAMS TM): Digital EMS

5. FUNDING NUMBERS
DAMD17-98-2-8002

6. AUTHOR(S)
James H. Duke, Jr., M.D.
James Wall, Ph.D., Jose Salinas, Ph.D., R. Douglas Tindall

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Texas Health Sciences Center
at Houston
Houston, Texas 77225-0708

E-MAIL:
duke@dreams-project.org

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army Medical Research and Materiel Command
Fort Detrick, Maryland 21702-5012

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 Words)*

Physician's virtual presence in support of the first responding care-givers at the scene of the incident will create the opportunity for achieving an accurate initial evaluation of the victim's clinical condition and a timely initiation of appropriate interventions. Any condition that interferes with adequate blood flow will cause an impairment of tissue oxygenation, the results of which are cell injury and, if sufficiently prolonged, cell death. The interval of time between the acute catastrophic events that initiate the decrease in blood flow and the establishment of therapies to reverse this cascade of cell injury is critical. Any measure that shortens the interval between the injury and the institution of appropriate therapy will afford the greatest potential for minimizing cell injury and preventing cell death. The DREAMS: Digital EMS project is designed specifically to address these issues.

14. SUBJECT TERMS	15. NUMBER OF PAGES
	104
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	Unlimited

FOREWORD

Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the U.S. Army.

Where copyrighted material is quoted, permission has been obtained to use such material.

Where material from documents designated for limited distribution is quoted, permission has been obtained to use the material.

Citations of commercial organizations and trade names in this report do not constitute an official Department of the Army endorsement or approval of the products or services of these organizations.

In conducting research using animals, the investigator(s) adhered to the "Guide for the Care and Use of Laboratory Animals," prepared by the Committee on Care and Use of Laboratory Animals of the Institute of Laboratory Animal Resources, National Research Council (NIH Publication No. 86-23, Revised 1985).

For the protection of human subjects, the investigator(s) have adhered to policies of applicable Federal Law 45 CFR 46.

In conducting research utilizing recombinant DNA, the investigator(s) adhered to NIH Guidelines for Research Involving Recombinant DNA Molecules.

In the conduct of research involving hazardous organisms, the investigator(s) adhered to the CDC-NIH Guidelines for Biosafety in Microbiological and Biomedical Laboratories.

Abstract

Digital EMS is one of the primary research areas under the Disaster Relief and Emergency Medical Services (DREAMS™) program. The main goal of the Digital EMS program is to increase the quality of emergency medical care in rural areas and on the battlefield by using high-quality video and data exchanges between the hospital and the EMS vehicle. By creating the virtual presence of a physician at or near the emergency scene, more lives will be saved because of the timely initiation of advanced lifesaving measures. The Digital EMS vehicle can also serve as a mobile command center in mass casualty situations.

This report serves as a review of the technical approaches implemented during the first year of The Texas A&M University System Digital EMS development, as subcontracted to the University of Texas Health Science Center at Houston. The overall technical architecture is described and specifications for equipment included in and modifications made to the ambulance prototype are detailed. Adequate detail is also provided to describe the user interfaces to provide insights into the overall operational capabilities of the Digital EMS technology.

Table of Contents

Table of Figures

Code Listings

1 Introduction

The Digital EMS segment of the DREAMS™ project is a two-fold effort for revolutionizing trauma-level care at mobile remote sites. The first goal of this effort is to create a virtual presence of a physician to assess emergency personnel during trauma-level emergencies. These include cases in which emergency personnel do not have the expertise or training to deal with the current situation. These emergencies involve not only typical EMS calls, but also could apply to military battlefield operations, remote rural hospital care, NASA space operations, offshore medical care, and air EMS. In these cases, patients may benefit from having physicians intervening at an earlier stage using this technology.

Multiple wireless and terrestrial communication means by provide high-quality video, audio, and data channels to transmit the necessary data back to an attending physician in a mobile environment. Emergency personnel at the remote site can be assisted directly by a physician and instructed on proper treatment. This allows for trauma facilities equipped with this system to virtually extend patient care to remote sites which would normally not be equipped to handle some types of trauma injuries.

The second goal of the project is to provide a system to assist emergency personnel in situations that may require triage support, such as disaster relief, domestic terrorist attacks, or other mass-casualty situations. Additionally, the connectivity provided by the system can be used to facilitate and enhance situation management. Digital EMS systems will be able to act as triage centers in which patients can be evaluated and prioritized by qualified physicians at a remote site. This is especially critical during situations in which large numbers of trauma patients require immediate treatment. Furthermore, this system can create a centralized control center and provide a point of presence at these sites with no setup time to enable communications into the affected area.

2 Digital EMS Technology Overview

Figure 2-1 shows the different technology areas in use currently or in the near future by the Digital EMS system on the mobile unit. These technologies include:

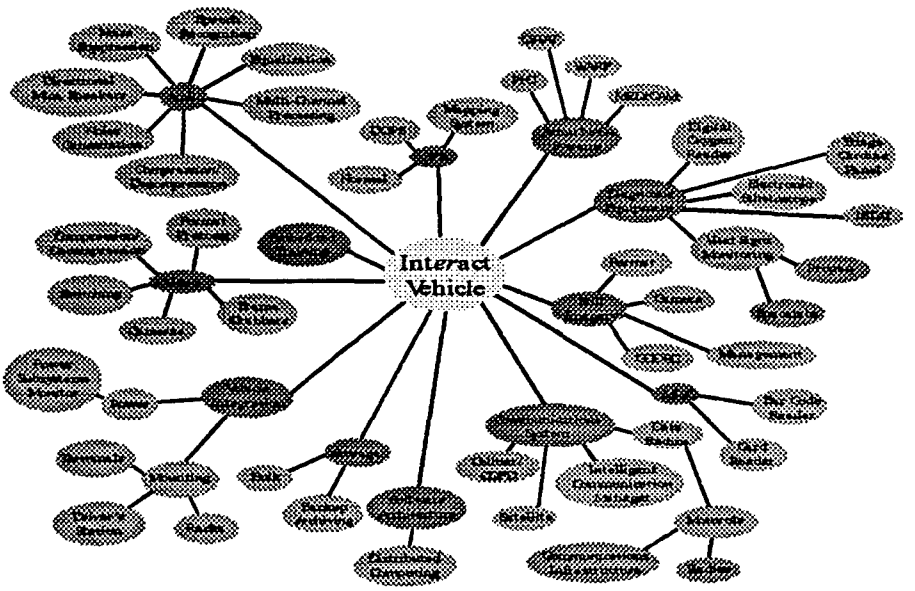


Figure 2-1. Technology areas covered by the project on the emergency vehicle.

- Video Processing

Current system design includes a set of video cameras mounted in the cabin of the emergency vehicle to provide patient images back to the hospital site. Additionally, a wearable camera is included to provide video data from the vehicle surroundings when worn by one of the vehicle emergency personnel.

- Hardware Integration

Vehicle systems must be modified to accommodate new hardware devices and systems, which must be integrated mechanically and electrically with minor modifications to the inside cabin specifications. Issues such as mounting, space requirements, and cooling become extremely important in an emergency vehicle high availability environment.

- Bulk Storage

When executing emergency calls, a mass storage system is required for backing up medical data, maintaining a record of the system status, and storing all data gathered by any system during the call.

- Software Architecture

System software design is based on a client/server architecture that requires multiple distributed applications to interface with each other and maintain a constant synchronized state. Additionally, a set of redundant and fault-tolerant procedures must be maintained for execution when problems occur during system operation.

- Wireless Communications

When dealing with the mobile environment of the emergency vehicle, wireless technologies included in the system must compensate for additional communication issues that arise during mobile data transmission and reception.

- High Resolution Imagery

In cases where the available bandwidth on the system is not large enough to sustain a video communication stream, high-resolution imagery can be used instead. In this case, issues that deal with different imagery aspects, such as formatting, compression, and image use protocols, must be taken into account.

- COTS Medical Diagnostic Equipment

Several commercial diagnostic devices have been or will be integrated into the system. Each device maintains a different interface that must be interfaced into the monitoring system.

- Government Diagnostic Equipment

In addition to commercial equipment, several government-developed devices will also be integrated into the system. Again, all device interfaces must be developed for use by the Digital EMS system.

- Mapping and Navigation

Location and route values must be maintained for each Digital EMS emergency vehicle in operation. This entails support for efficient route computations on board the system that can assist the driver with better navigation during emergency calls.

- Audio Processing

In addition to video needs, physicians at the hospital must be able to communicate with the vehicle using clear and understandable dialog. Hence, issues of format, compression, and transmission of voice data must be taken into account, as well as technologies for suppressing vehicle environment noise.

- Embedded Training

Finally, the system includes an embedded training requirement for vehicle staff to refresh training if needed when not on an emergency call.

Figure 2-2 shows the technologies impacted by the Digital EMS system on the hospital site. These technologies include:

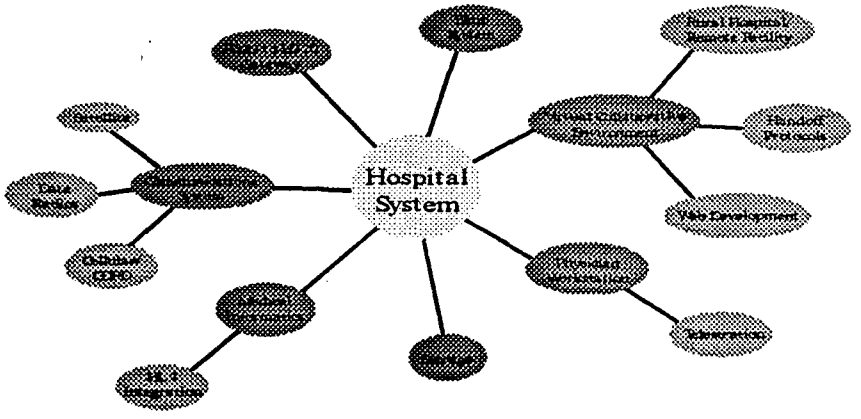


Figure 2-2. Hospital side technology areas covered by project.

- Wireless Communications

Some of the same issues that the mobile environment has with the wireless systems will also be present in the fixed hospital station.

- Collaborative Environment

The hospital station requires that other systems in the hospital be able to log in and participate or assist during a session with the remote vehicle. This involves management of multiple control stations and requires the development of handoff protocols for transferring control to other physicians in the system which may better assist the emergency vehicle staff.

- Video Conferencing Formats

H323 will be implemented in the system for establishing collaboration sessions required by multiple remote logins. However, this format is an umbrella standard that encompasses other standards for different collaboration aspects. Formats for video compression, data transfer, voice compression, etc., are mandated by H323, but in some cases alternate technologies will be investigated. Digital EMS will implement additional, more advanced compression methods.

- Medical Informatics

Patient records generated by the emergency staff are ultimately managed and processed by Medical Informatics protocols outside the Digital EMS system. However, current medical informatics standards do not support all data management requirements for the Digital EMS

system. Recent work is focused on using the HL7 standard for transferring patient records from/to the hospital databases to/from the Digital EMS system; however, this standard does not meet the multimedia transfer requirements generated by the emergency vehicle during a call with an active video system.

- User Interface

Finally, interface issues are currently being examined for creating a more intuitive user input into the system that minimizes user interaction. This will allow personnel at both ends of the connection to concentrate on the patient treatment, while minimizing interactions with the system at all times.

3 System Software Architecture

The current system implementation consists of a set of computer systems interconnected by a high-speed Ethernet connection within the emergency vehicle and connected to additional systems using an Ethernet wireless LAN connection. Each computer provides a subset of the total functional requirements necessary for full system functionality. The software applications executed on the system are fully distributed among the three computers to provide the necessary interfaces to the emergency personnel, system hardware, and attending physicians using the system.

The existing software design of the Digital EMS system consists of a set of client and server modules with a tightly coupled communications scheme. Currently, communications across different applications is achieved through the use of Remote Methods (RMI) which allow clients in the system to call and execute object methods implemented on a server application in a distributed system. This architecture defines a distributed approach for implementing the different functional aspects required by the Digital EMS system. Figure 3-1 shows client/server communication using the Java RMI Applications Protocol Interface (API).

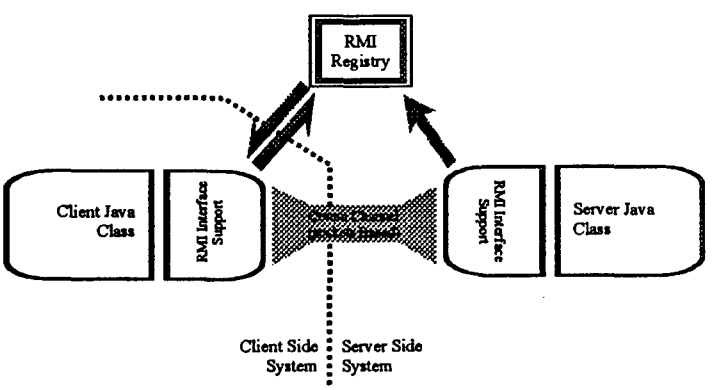


Figure 3-1. Client/Server communications via Java RMI.

One of the drawbacks of using an RMI approach for communications is the overhead associated with the serialization and transfer of parameters and results across the network. This becomes a critical point when dealing with the bandwidth-limited communication system within the Digital EMS emergency vehicle. Additionally, the RMI implementation will abstract the communication details away from the software developer by automating the serialization and transfer of objects across the network. This will not allow for a customized communication protocol as required by the ambulance/hospital system. Hence,

client/server communications are currently being rewritten to use a socket-based approach as shown in Figure 3-2.

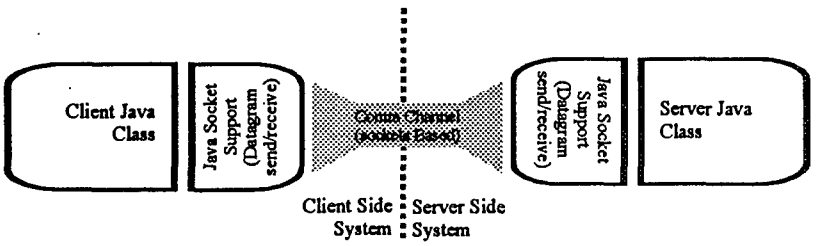


Figure 3-2. Client/Server communications via a socket based infrastructure.

One of the disadvantages of using a straight socket implementation is the lack of control over client/server packets below the application layer. This becomes a critical problem when dealing with a mobile system that maintains a set of multiple heterogeneous communication devices, any of them capable of supporting a section of the overall operational communications bandwidth for all applications executing in the system. Work is currently in progress to implement a middleware layer which will manage client/server communications transparently to the applications using an Intelligent Communications Manager (ICM) tool for facilitating data transfers across multiple devices. Please refer to Section 12.6 for a description of this ongoing work.

4 Software Development Environment

Software development for the Digital EMS system is mainly done using the Java 1.2 programming language through the use of the Java Development Kit 1.2.2 (JDK 1.2.2), which is distributed through an open-source license. This language has been standardized at the software definition and the compiled (byte-code) level. Hence, this environment will ensure software portability across multiple platforms that support the language definition. In this manner, applications developed using this environment will be fully Java compliant and portable across any platform that implements the Java Virtual Machine (JVM 1.2.2) definition and Run Time environment standard.

All software modules in the system have been either written completely in the Java 1.2 standard or have a Java "wrapper" which encapsulates the module's capabilities in a Java framework that can be accessed by other system applications. With the exception of video display and sound play objects, all clients in the system are written using the standard Java Swing Interface for GUI development. Servers in the system use Java if their functionality is supported fully by the language (or any of the standard Java extensions). Otherwise, the server functionality is written in C++ and is encapsulated by the Java Native Interface (JNI).

4.1 Additional Application Protocol Interfaces (APIs)

Apart from the core Application Protocol Interface (API) provided by the JDK, two additional Java APIs are used by some of the Digital EMS modules for enhanced functionality including:

- Java serial communications API (JavaComm).
- Java database access API (JDBC).

4.1.1 JavaComm

The *JavaComm* API (not part of the core Java API in the JDK 1.2.2) provides the necessary functionality for accessing, controlling, reading, and writing data from RS-232 serial ports on a PC under Microsoft Windows (9x and NT) OS, Solaris OS, and Linux OS. This API provides a standard set of class objects that can be cast into standard Java IO streams for reading and writing serial data from all serial ports in the system. Ports can be fully controlled by a set of serial port objects that access the underlying hardware through a set of serial object access functions including:

- Enumerating the system ports.
- Setting the port name.
- Changing the port baud rate.
- Setting the flow control of the port.
- Setting the data bits and stop bits of input data.
- Setting the port parity.
- Checking port ownership.
- Claiming port ownership.

Data is sent/received form a port by creating a serial port object that includes, as member functions, a set of write and read methods that can be cast into standard Java IO streams. Data is sent to the port by writing to the serial object once the port has been initialized. Reading data from the port can be done in a polled or event driven fashion depending on the amount of data expected from the port.

In a polled situation, the application program will check for data availability periodically until the data available flag is set on the port status registers. Data can then be read by executing a write on the port. In an event driven situation, the application registers for *SerialPortEvent* from the Java serial port event queue and implements a *SerialPort* interface. When data is received, a *SerialPortEvent* is generated which will call the *SerialAction* method (defined by the interface) in the interested application. Data reads can then be done in this method until the port queue is empty.

4.1.2 Java Database Access API (JDBC)

This API provides the necessary connectivity for accessing local or remote databases in a database and platform independent manner. It provides a call level API for connecting and executing SQL level queries on databases which support standard SQL statements. Databases can be accessed by the API using four different approaches:

- JDBC-ODBC bridge plus ODBC driver.
- Native JDBC API.
- Direct to database access driver.
- Pure Java Driver for Database Middleware.

This API is being further developed to access Patient Record information to assist emergency personnel in filling out the Emergency Run Record.

4.2 The Java Native Interface (JNI)

Hardware dependent functionality is accessed in the system by the use of the Java Native Interface (JNI). This API allows Java objects to call functions and methods that have been written and compiled in other languages. In this manner, software that depends on non-Java development libraries or requires specific calls to hardware-dependent functions (such as specialized operating system functions) can be written and compiled using a native compiler for a specific system. These native functions are compiled into native libraries which can be called from Java objects that have these functions declared as native methods within the calling class.

Digital EMS libraries using the JNI functionality include:

- Frame grabber software.
- Video display functions for Paramedic-Mobile modules.
- Video and thumbnail display functions for Physician modules.
- Sound server functions for recording microphone.
- Sound client functions for sound playback.

5 Package Interface and CLASSPATH Settings

The current software architecture uses the Java packages' organizational structure to locate the different system modules during run-time execution. All packages for the system are located under the main *dreams* package in the root directory (`C:\Dreams`) of each machine executing the software. Clients, servers, and supporting Java classes implement the *dreams.package* interface (where *package* is the role of the module – i.e. *videoserver*). In this manner, the `$CLASSPATH` variable needed by the JVM executable needs to only to be set to the root directory of the main package location (i.e. `$CLASSPATH=C:\`). All class files in the system can then be located by the JVM in the `C:\Dreams\Module` directory.

Figures 5-1 and 5-2 show all the packages used for the DREAMS software modules. The system uses seven main server packages during run time. These provide all functionality required for accessing all data managers in the system that send and receive data and/or provide hardware access and control to different system functions.

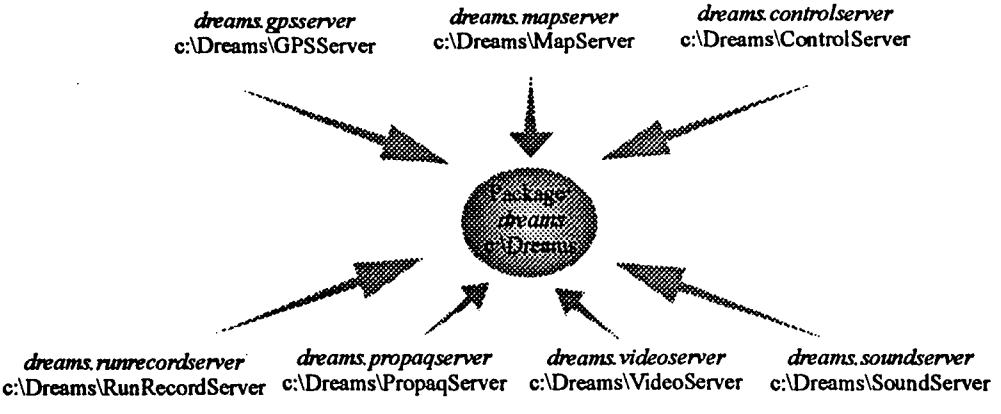


Figure 5-1. Server packages for the Digital EMS software modules.

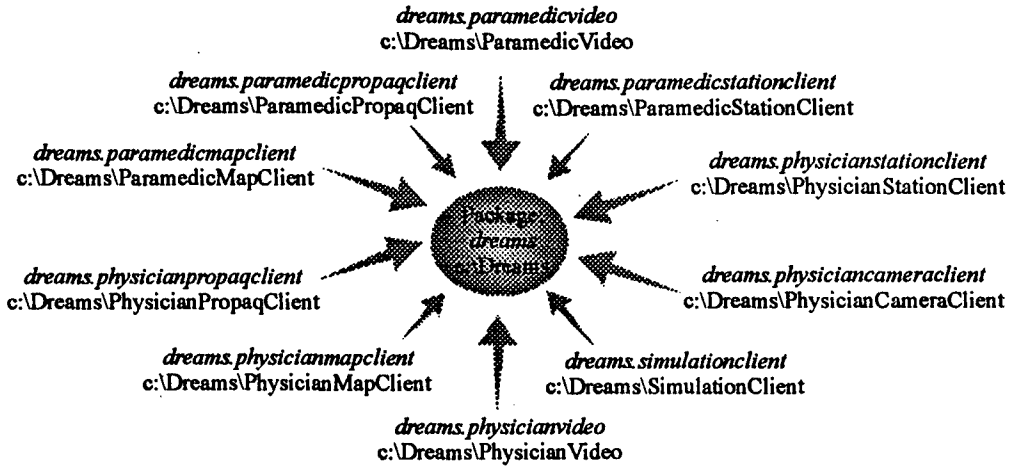


Figure 5-2. Client packages for the Digital EMS software modules.

The client packages provide the set of class files for running and executing the different clients in the system. These include modules for interfacing to the emergency personnel in the ambulance and attending physicians at the hospital site. Additional directories' modules are provided under the main `C:\Dreams` tree (i.e. `C:\Dreams\VideoServerSock`) which replace existing modules and provide additional modes of operation for other run-time modes. These are renamed to match the package interface that they provide in order to be used by the system at run time.

6 Operating System Considerations

Modules executing on the Digital EMS system constitute a set of tightly coupled applications that make up a closed system. Portability across multiple platforms allows for many modules to execute on different machines within the system. Consequently, the operating systems chosen for this project are required to support distributed programming techniques that allow application network communications in a fast and efficient manner.

The operating systems chosen for development and implementation include Microsoft Windows NT version 4.0 and the Linux operating system. All client/server modules and applications for the Digital EMS system are implemented for the NT environment. However, modules that make up the communications system are written in C++ for the Linux operating system. This OS is a Unix-like POSIX-compliant system with full support for multitasking and distributed development. It was chosen due to the large support base and source code availability of communication applications (including operating system source code) and straightforward software development environment.

7 Run-Time Environment

The Digital EMS system is initialized in a two-phase manner which allows for software applications in the distributed architecture to register themselves with the system, identify other run-time objects, and communicate with other modules in remote systems through the Digital EMS communications infrastructure. The first phase consists of executing the Java RMI registry tools on any machine that will have one or more servers loaded on it. This will create a set of system registries that are used by the Digital EMS clients to locate and access particular servers in the system. The second phase will execute

the client and server modules to initiate the application run-time environments, register applications with the system, and commence client/server communications. Servers initialize themselves by creating a server object, which registers itself with the system and then waits for client connections. Once all servers are registered, clients on other machines can use the registry for locating and accessing the required server functionality across a distributed architecture through the Java RMI method invocation protocol.

Software modules are mapped into the three machines that currently make up the overall Digital EMS architecture. Two computers make up the emergency vehicle subsystem (*ambulace1.tamu.edu* and *ambulance2.tamu.edu*), while the third machine implements the hospital side functionality. Modules executing in the emergency vehicle include all of the video processing and control applications, the medical device interfacing modules, and the emergency personnel interface applications. Figure 7-1 shows the mapping of these software modules into the two systems in the emergency vehicle.

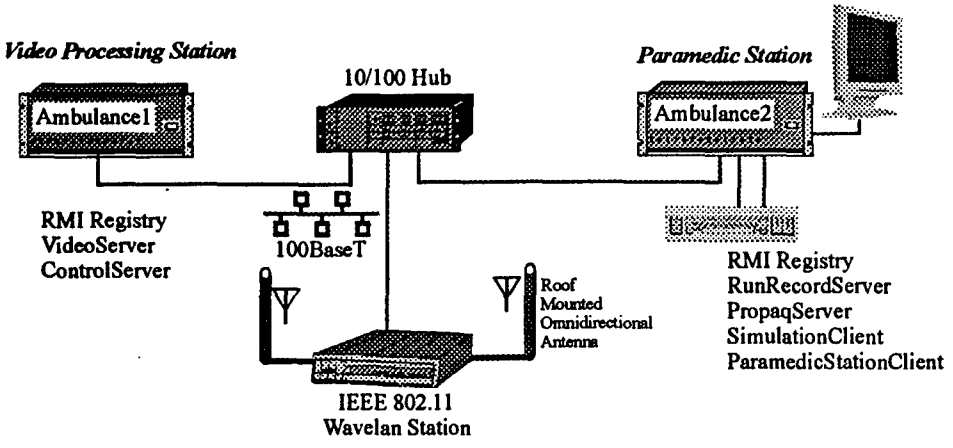


Figure 7-1. Software module mapping into emergency vehicle systems.

The video and processing station (Ambulance1) executes the Video Server for processing the on-board cameras and Medic-Cam video and executes the Control Server for controlling the PTZ (Pan, Tilt, Zoom) functionality of the patient monitoring cameras at the request of system clients. The second system (Ambulance2) implements the modules for accessing the medical devices and interfacing with the EMS personnel on board. These include the Propaq Server vital signs monitor, the Simulation Client for initiating demo runs, the Run Record Server for managing patient record information, and the Paramedic-Mobile Station Client for providing the interface to the EMS personnel in the vehicle.

The applications currently executed on the Physician Station are the two clients for interfacing with the attending doctor. Figure 7-2 shows the hospital-based system with the mapped software modules during run time execution.

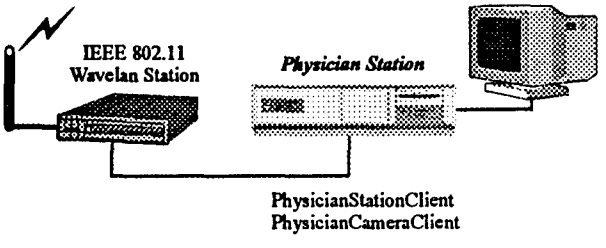


Figure 7-2. Software module mapping into hospital side workstation.

The Physician Camera Client will provide the video display and PTZ interface control of the patient cameras in the remote vehicle, including the camera presets and thumbnail displays. Additionally, a Physician Station Client is executed in a separate window for implementing the vital signs display and control, the run record display, and fleet management mapping and navigation interface to the attending physician.

Each computer is equipped with a set of batch files that, when executing the Digital EMS software, will execute the predefined set of applications for each machine in the system. This allows for automatic initialization of the system during startup or during manual configuration.

8 System Clients

Clients are defined as those modules which interface with the software servers and/or users to provide the front end functionality of the system. Currently, the system is composed of four main clients:

- Physician Camera Client.
- Physician Station Client.
- Paramedic-Mobile Station Client.
- Sound Client.

Clients in the system are configured by a set of local configuration files that are read-in during client initialization. Each client requires a configuration file in the local directory where the main package is located. Configuration files contain run-time parameters required by the clients to correctly set up the run-time execution environment of the client. Parameters include:

- Required server locations.
- Com ports where devices are located.
- User preferences.
- Any other client-specific parameters.

The format for the configuration file is given as follows:

- Parameter1=Value
- Parameter2=Value
- ...

with # characters and blank lines given as comments. Listing 8-1 shows an example configuration file for the Paramedic-Mobile Station Client.

```
# this is the configuration file for Paramedic Station Client

PropaServer=digitalems2.tamu.edu
VideoServer=digitalems1.tamu.edu
GPSServer=digitalems1.tamu.edu
MAPServer=digitalems2.tamu.edu
```

Listing 8-1. Example configuration file for Paramedic-Mobile Station Client.

Each client in the system maintains a parser object for reading and processing its local configuration file during the client initialization.

8.1 Physician Camera Client

This client is responsible for controlling the cameras on the emergency vehicle (including the Medic-Cam), displaying the video streams from the Video Server, and interfacing with the attending physician at the hospital site. This client is executed on the Physician Station at the hospital side when the system is initialized and constitutes the first pane of the physician interface. Servers supported by this client include the Video and Control servers that are used for streaming the video data from the selected camera and controlling the cabin camera movements. Figure A-1 (Appendix A) shows the dependency diagram for the main Physician Camera Client Java class.

This client depends on the interfaces from the two supporting servers and Stub file for each server for establishing the communications infrastructure to create the RMI links. Additionally, this client uses the JNI interface for painting the video to the screen. Under the standard Java 1.2.2 drawing interface, the JVM is not fast enough to draw video at 30 fps on a 640 x 480 display window. Hence, the video display was implemented using the Microsoft DirectX interface. This API allows applications to access the video memory and display buffers directly to achieve the required performance when displaying full-motion video. Video display support is implemented by the *PhysicianDisplay.dll* file that contains the JNI interface into the DirectX methods located in the *Ddraw.dll* library.

Video is received from the Video Server using server pull protocol to extract individual frames from the Physician Station frame grabbers. Frames received from the server are compressed in a Motion Wavelets format with an attached header description. Incoming frames are transferred to the decompression system implemented in the PhysicianDisplay.dll library. Each frame will be decompressed to the required resolution with an RGB 565 format. This decompressed frame will then be given to the DirectX system for painting to the region of the display where the video is presented.

8.2 Physician Station Client

This client constitutes the second display (along with the Physician Camera Client) which is used by the attending physician at the Physician Station system to control the Propaq Vital Signs monitor, implement the fleet management mapping interface, and display a non-modifiable copy of the Emergency Run Records being modified in the ambulance. Three panels are used for each of the client functions with an associated push button to switch between them. Figure A-2 (Appendix A) shows the dependency diagram for the Physician Station Client Java class.

This client is supported by five servers executing on the remote ambulance, including: the Propaq Server, Run Record Server, GPS Server, and Mapping Server.

8.3 Paramedic-Mobile Station Client

Personnel in the emergency vehicle primarily use this client as the main application for interfacing into the system. The client is executed on the Paramedic-Mobile Station and is composed of seven different panels personnel can use to access the different system functions including:

- Automated Emergency Run Record Management.
- Video/Vitals Display.
- Emergency Protocols.
- Mapping and Navigation Interface.
- Communications Display.
- Run Record Report.
- Training.

EMS personnel use the first panel for creating and managing the Emergency Run Records required during an emergency call. Run records created and modified in this panel are automatically transferred to the Physician Station through the Run Record Server. A second report panel is also available for non-critical information input. The Video/Vitals display is used for displaying the same video stream that is currently being painted on the Physician Station. This allows for a common frame of reference between the Paramedic-Mobile and Physician stations. Similarly, the current vital signs being read in by the Propaq Server are also displayed this panel. Emergency protocols are displayed on the protocols panel using HTML format. This allows for easy navigation of the different protocols through the use of familiar web tools and methodology.

The mapping and navigation system is used for map location of the unit and route planning from the vehicle's current location to the site of the call and from the call site to the destination hospital. The communications display is used for checking the status of the communication channels and modifying communication parameters for mounted devices. Finally, the training panel is used for emergency procedure training of emergency personnel.

This client depends on five servers including the Video Server, Propaq Server, Mapping Server, and GPS server. Video is displayed using the Microsoft Direct X API to achieve the required frames per second in the display. When users switch to non-video panels, the system contains a set of methods for stopping and clearing the screen before the next panel is displayed. In this manner, video frames will not be written to the screen in non-video panels and video server performance will increase on other clients displaying video.

When displaying video on the video/vitals panel, the Motion Wavelets system decompresses the incoming image from the Video Server. The real-time compression engine at the server side compresses frames extracted from the Video Server. These frames are transferred to the Motion Wavelets decompression system for extracting the full size image to display on the screen. Images are extracted in RGB-565 format, which includes a 16 bit display depth with a six-bit green color.

Vital sign values are extracted in the same manner as the Physician Station Client and are displayed on the same panel as the video stream. This will create a screen that will have a common frame of reference for both the physician and vehicle personnel during a collaboration session.

Figure A-3 (Appendix A) show the dependency diagram for the main Paramedic-Mobile Station Client Java class.

8.4 Sound Client

The sound client plays back sound frames extracted from the Sound Server. The client uses the server pull protocol to control when frames are transferred from the server to the client. Once a packet has been

received, the client uses the Microsoft Direct Sound interface to open the local playback device and play the sound packet.

Figure A-4 (Appendix A) shows the dependency diagram for the main Sound Client Java class.

This client does not implement a user interface during execution, so it can be executed in the background with the main Paramedic-Mobile and Physician clients. Additionally, this client only implements a half-duplex connection to the remote Sound Server. In order to establish a full duplex connection between the vehicle and the hospital site, two sets of client/server pairs are executed for creating the two links between stations, as shown in Figure 8.4-1.

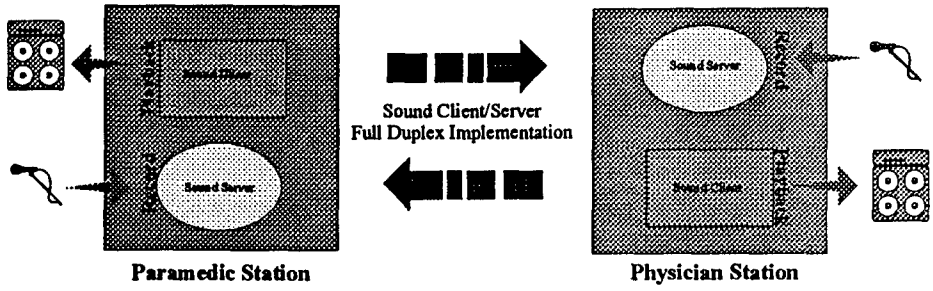


Figure 8.4-1. Sound Client/Server full duplex implementation.

9 System Servers

Servers in the DREAMS system are defined as software modules that either manage data and/or interface with system devices on behalf of clients in the system. In the case of the Digital EMS software, seven servers have been implemented for providing functionality to different aspects of the data and hardware within the system including:

- Video Server (package *dreams.videoserver*).
- Control Server (package *dreams.controlserver*).
- Sound Server (package *dreams.soundserver*).
- Propaq Server (package *dreams.propaqserver*).
- Run Record Server (package *dreams.runrecordserver*).
- GPS Server (package *dreams.gpsserver*).
- Mapping Server (package *dreams.mapserver*).

9.1 Video Server

The Video Server is executed within the Video Processing Station inside the emergency vehicle and is used for interfacing with the video hardware. This includes accessing the frame grabber hardware within the system, interfacing with the cameras in the system, compressing video frames for sending to requesting clients, and managing multiple video and thumbnail streams from multiple sources. Figure 9.1-1 shows the different components required by the server.

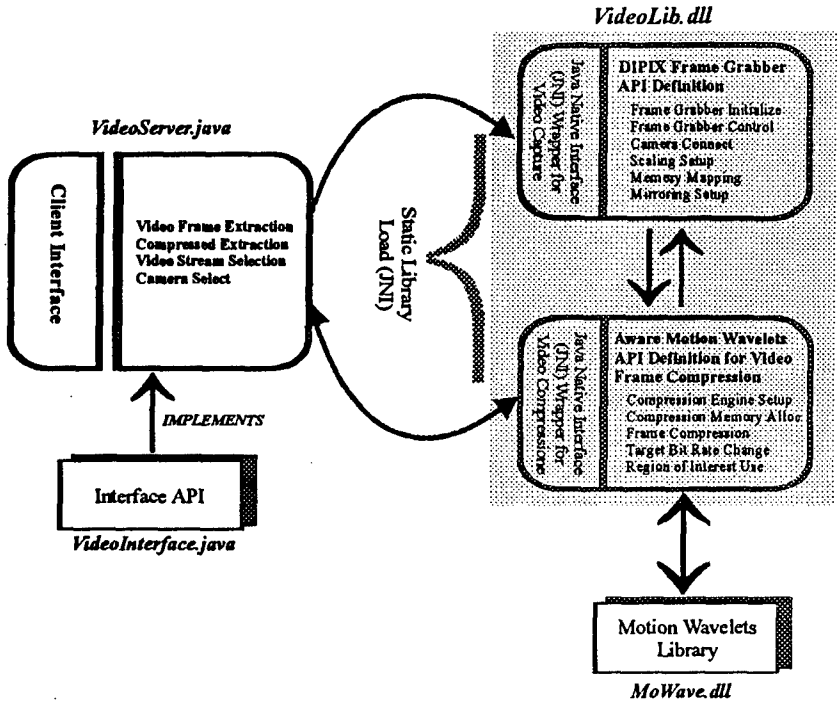


Figure 9.1-1. Video Server components.

The server configuration is composed of three main components including:

- Video Server main program.
- Interface API.
- Library DLL.

The main program is comprised of the implementation functions for the API defined in the interface module. This includes all the functions available to the clients in the system and any internal functions required by the server during execution. A *VideoLib.dll* library is statically loaded by the server during initialization and is used to access the frame grabber and compression functionality not supported directly by the Java language.

Video data is sent as compressed frames to requesting clients in a pull configuration. Clients in the system that need to display or process video data interface with the Video Server via a set of requesting functions for single compressed video frames. Requests are composed in the form of compressed frame requests to the server with an optional camera number. A pull configuration requires that frame transfers to the requesting client be achieved at the control of the clients themselves. Once a request has been received for video data, the server will return a single frame to the requesting client and end the transaction. Requests for additional video frames require additional video frame requests by each client. Figure 9.1-2 shows the flow diagram for interfacing from a requesting client to the server for a compressed frame.

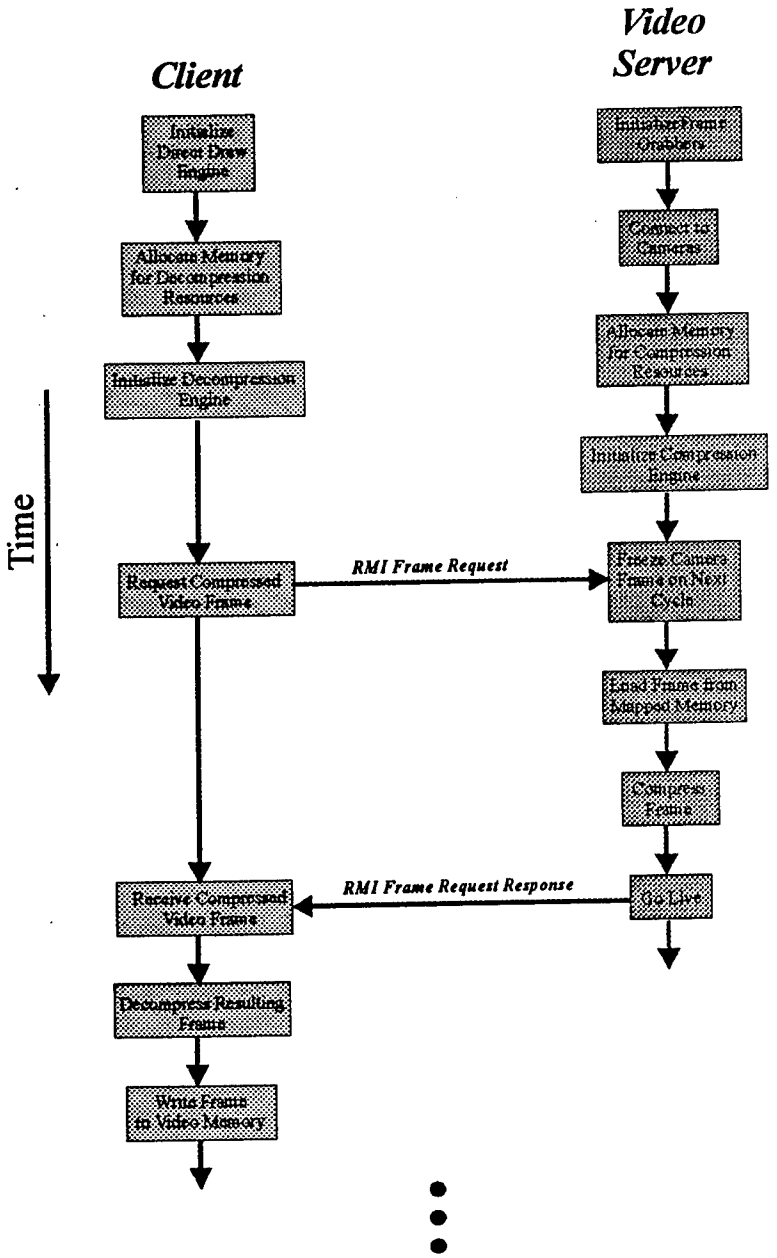


Figure 9.1-2. Video Server communications protocol.

Execution of the server is accomplished by the main program in the *VideoServer.class* file. This function will create a *VideoServer* object that will be registered with the local RMI registry in the system. This name will be used by clients requesting connections to the server from other machines in the Digital EMS network. Once the server object has been created and registered with the system, a set of native initialization routines are executed on the *VideoLib.dll* library for setting up the video and compression system. Initialization is composed of a two-phase approach, which includes:

- Camera and Frame Grabber Initialization

This phase will initialize the frame grabbers in the system and allocate memory for video storage. Initialization includes the following steps:

- Find the grabbers in the system (up to 4 simultaneous grabbers are supported).
- Connect the cameras attached to the grabbers. Cameras attached to the grabbers can be connected on either an NTSC of SVGA connectors on each grabber. However, camera connections are limited to the NTSC connector for the Digital EMS software.
- Allocate memory for grabber and camera control information.
- Set the Region of Interest for each grabber.
- Choose the video input format to the standard YUV 4:2:2.
- Set the dimensions of the output image.
- Define the memory mapping for captured video locations in main memory.
- All grabbers not connected to a Medic-Cam system will be set to mirror the image upside down to account for the camera mounting on the roof of the ambulance.

▪ Compression System Initialization

This phase will initialize the compression system and allocate memory for storing compressed frames. A set of four compressors is initialized simultaneously for compressing from each grabber independently. All compressors are initialized with a shared memory option to improve performance and reduce run-time memory requirements for the server. Parameters for compression are given during this initialization to determine the compression rate and bandwidth requirements for the video stream.

Once the server initialization phase is complete, clients requiring video can start requesting compressed frames from the system. When a client request for a video frame is received, the server will first freeze the camera from which the frame will be captured. This will freeze the current image in the grabber memory and prevents the server from having overlapping frames during frame transfers to requesting clients. A camera freeze is done on scan boundaries to prevent the camera from freezing before completing the image digitized from the camera inputs.

Once a full frame is available from memory, the server will call the compressor for that grabber to extract the camera image located at the mapped memory location and compress it based on the given set of compression parameters. The resulting compressed image is returned as a memory location and size which is then copied into a result byte array and sent back to the requesting client. The camera is then unfrozen and returned to a live mode for further requests.

When no camera number is specified during a client request, then video will be extracted from the current camera in the system. Calls are also provided for querying the server for the current video resolution of the frame grabbers attached to the server for setup of local client parameters. Resolution can be queried for horizontal and vertical width in pixels and pixel depth in bits per pixel.

9.1.1 Video Interface

The Video Interface consists of the set of calls implemented by the video server and available to remote clients. Function calls are defined within the file using the Java *interface* definition. This allows for calls to be defined but not implemented and is similar to function prototypes in the C and C++ languages. For a client/server implementation, the interface is used by the clients to create objects that contain the functions available in the server. On the server side, the main server class will implement the Video Interface and create the bodies of the functions listed in the interface definition. Listing 9.1.1-1 shows the Video Server interface definition.

```
package dreams.videoserver;
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface VideoInterface extends Remote {

        public short   getVideoHeight(int CameraIndex ) throws RemoteException;
        public short   getVideoWidth(int CameraIndex ) throws RemoteException;
        public short   getVideoPixelDepth(int CameraIndex ) throws RemoteException;
        public byte[]  getVideoFrame(int CameraIndex) throws RemoteException;
        public byte[]  getVideoFrame1() throws RemoteException;
        public byte[]  getVideoFrame2() throws RemoteException;
        public byte[]  getVideoFrame3() throws RemoteException;
        public byte[]  getVideoFrame4() throws RemoteException
        public short   getVideoHeight() throws RemoteException;
        public short   getVideoWidth() throws RemoteException;
        public short   getVideoPixelDepth() throws RemoteException;
        public byte[]  getVideoFrame() throws RemoteException;
        public void    setCurrentCamera(int CameraIndex) throws RemoteException;
        public byte    getCurrentCamera() throws RemoteException;
        public byte    getNoCameras() throws RemoteException;
```

Listing 9.1.1-1. Video Server interface definition.

The four *getVideoFrame*()* functions are used for extracting the compressed video frames from the server for each of the four cameras supported. Each frame will be transmitted directly to the client in a compressed Motion Wavelets format with an attached header for frame description. The other functions in the interface are used for querying and setting different parameters of the server during execution.

9.1.2 Frame Grabber Interface Software

Interfacing and controlling the attached Dipix PicPort Color frame grabbers is done using the Leutron LVS-DS real time library. This library interfaces with the grabbers by creating a system of interconnected nodes representing the underlying hardware attached to the system. This structure can be used for accessing and controlling the different hardware settings for all the attached grabbers including the flow of captured data within the different system paths.

Each of the nodes in the system represents either a Camera or Grabber node. Camera nodes are used for controlling the different aspects of the attached cameras, whereas the Grabber nodes control the grabber settings. Applications access the system using the set of node access functions to change the required parameters. The *DSY32.dll* library is used to handle these settings during run time.

9.1.3 Wavelet Compression System and Interface

Video compression in the server is achieved through the use of the Motion Wavelets real-time video compression system. This system provides a set of routines for compression of video frames in real time using a wavelet compression scheme on a frame-by-frame basis. Implementation of the system is done through the use of a set of high-performance routines that execute on MMX-enhanced Pentium class processors. The system uses a reentrant DLL for supporting multiple simultaneous compression and decompression calls with a theoretical limit of > 60 fps on a 320 x 240 image size and 30 fps on 640 x 480 images. The compression algorithm uses only intraframe compression, which increases the

compression efficiency when streaming over low-quality communication systems which may lose frames during transmission.

Compressing a video frame consists of a three-step process that includes:

- Compressor Initialization

One compressor per video stream is recommended for maximum compression performance.

- Frame Compress

Using the set of supported compression routines, a frame is compressed based on the given initialization parameters. This step is repeated multiple times based on the number of frames to compress.

- Compressor Deallocation

The system should release resources allocated for the compression mechanism. This also entails releasing memory allocated for storage of compressed and raw image files.

9.1.4 Video Server dependency diagram

Figure A-5 (Appendix A) shows the video server dependency diagram for the current implementation of the server Java classes.

The main class in the Video Server package is the *VideoServer.class* object. This class is responsible for registration of the server with the RMI registry, implementation of the interface functions and interfacing with the DLL files through the use of native functions. Clients in the system must include the *Stub* class generated from the RMI compiler for the Video Server in order to communicate with the executable. This file is generated along with the *Skel* file during RMI compilation and is used internally by the JVM to implement the client/server communications.

9.2 Control Server

The Control Server is also executed within the Video Processing Station inside the Digital EMS emergency vehicle along with the Video Server. This server is used for interfacing with the camera control system to access the movement and zoom functionality. Camera control is interfaced into the system through the set of RS-232 ports in the Video Processing Station. Each camera is connected to the serial port that matches the frame grabber number inside the system's PCI bus. Camera control functionality is accessed using the VISCA/RS-232 camera control standard. This standard implements a Video System Control Application standard for implementing a serial communications bus interface that enables connection of up to seven devices in a daisy chain. Figure 9.2-1 shows the main components of the Control Server.

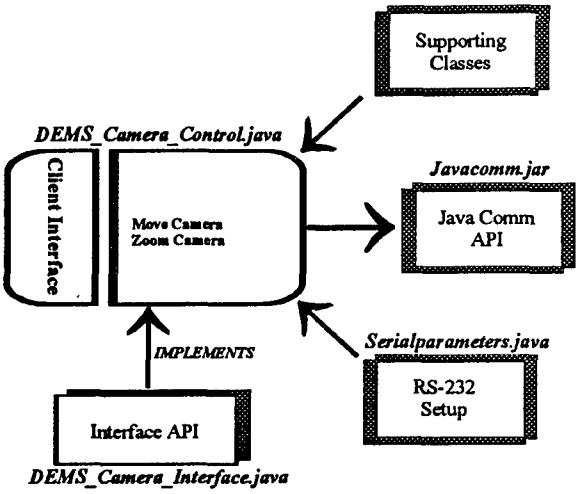


Figure 9.2-1. Control Server components in Digital EMS system.

The two main functions implemented by the Control Server are the Move and Zoom commands. By giving the direction of the move, the camera implements both the Pan and Tilt capabilities. Camera zooming is done by the Zoom command by giving a magnification parameter when called.

Clients interface with the Control Server using the same approach as the Video Server interface. Listing 9.2-1 shows the interface implemented by the clients to access the Move and Zoom functions of the server. By giving the camera number in the request, the server will choose either *COM1* or *COM2* for sending the VISCA command.

```
package dreams.controlserver;
import java.rmi.Remote;
import java.rmi.RemoteException;
public interface DEMS_Camera_Interface extends Remote {
  public void Move_Camera (int camera, String Direction) throws RemoteException;
  public void Zoom_Camera (int camera, String Factor) throws RemoteException;
```

Listing 9.2-1. Control Server interface definition.

VISCA commands sent to the local RS-232 serial port are transferred using the *JavaComm* API without the need for implementation of a native call interface. Presets for the cameras are stored on the client side and are sent to the server via the Move and Zoom commands.

Camera movement speed for each camera is currently fixed to a value of one (1), which moves the camera at the slowest speed available. This allows for better fine-tuning of the camera positioning inside the Digital EMS vehicle. Additionally, by minimizing the movement speed, delays in communication to the vehicle will not affect the fine control of the cameras from a remote site by having users overcompensate during communication non-responsive lapses. In the case of zoom speed, the speed is fixed at a value of four (4), which defines the middle zoom speed. For this case, fine control is not as noticeable when zooming the camera from a remote location; hence the value is fixed at a higher rate.

The main class in the Control Server package is implemented by the DEMS_Camera_Server.class file. Figure A-6 (Appendix A) shows the dependency diagram for this class.

The main class in the server depends on four classes that implement the server interface, the communication skeleton, server testing, and an exception class. A *Stub* class is also generated during the RMI compilation process, however, the dependency for this class is on the client side. Since this server is using the *JavaComm* API, the CLASSPATH will also need to point to the *comm.jar* file. This file contains the serial communication classes for this API.

9.3 Propaq Server

This server is used for managing and interfacing with the Propaq Vital Signs Monitor. This monitor is used during transport to monitor multiple patient vital signs using a set of sensors attached to the patient.

Interfacing with the monitor is done using one of the RS-232 ports available on the Paramedic-Mobile Station system on-board the emergency vehicle. Once a patient has been loaded into the vehicle, power and data connections are attached to the monitor to initiate data transfers between the server and the monitor data port.

Figure 9.3-1 shows the different components of the current Propaq Server implementation. This implementation consists of the push type server that provides an interface between a set of client in the system and the monitor through the use of control and data packets defined locally.

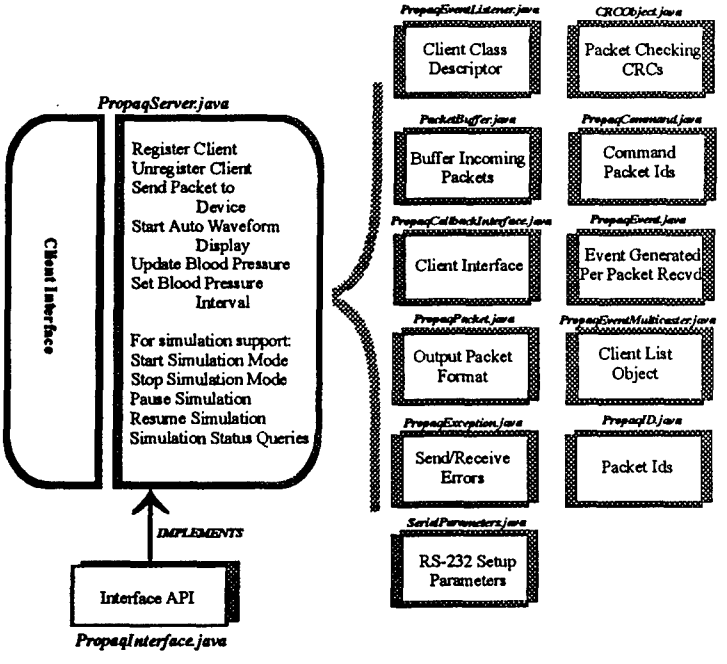


Figure 9.3-1. Propaq Server components in system.

The main class of the server is the *PropaqServer.class* file. It is currently supported by a set of additional objects that implement the required functionality of the server including:

- Sending/Receiving packets to monitor.
- Registering client for numeric and waveform packets.
- Serial port communications.
- Propaq event generation.

- Client push protocol.
- Simulation support.

An initialization phase is performed during server startup to create an initial data link between the server and the monitor. This establishes the necessary data link connection for either interactive or continuous data acquisition from the monitor. Data link initialization is accomplished by sending a Control Initial Byte Sequence to the monitor port. This sequence contains a set of six bytes given by:

- {0x02, 0xfd, 0x08, 0x01, 0x09, 0x00}

Communication is performed by sending/receiving Propaq packets to/from the monitor. Packets have a standard format, which include a header, data, and error detection sections as shown in Figure 9.3-2.

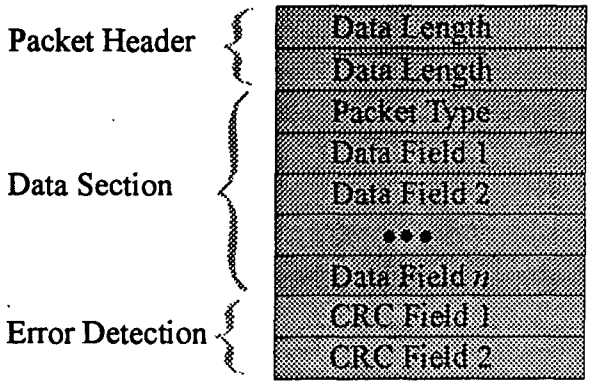


Figure 9.3-2. Propaq packet definition.

The packet header section constitutes the first two bytes of a Propaq packet and consists of the length of the packet in the first byte and the binary complement of that number on the second byte. The packet data section has the Propaq packet type and any data fields associated with the packet type. The last type bytes of the packet are used for error detection and constitute the CRC field of the previous packet bytes. Packet types are subdivided into four types of categories including:

- System Command Packets (0x0_)

These packets are used for sending commands for changing system configuration and/or system control parameters to the monitor.

- Request Packets (0x1_)

These packets are sent from the control application to the monitor for requesting data values from the different on-board sensor attached to the patient.

- Response Packets (0x2_)

These packets are sent from the monitor to the application when responding to requests from the application. These packets are only sent as a response to a previous request.

- Auto Packets (0x3_)

These packets are sent on a periodic basis to the requesting application whenever the monitor has been set on automatic mode. Waveform data is transferred to the application every 88 msecs with a set of values corresponding to one of the requested monitor waveforms. Numeric data is transferred to the application every two seconds for updated values for the vital signs.

Clients interface with the server by registering for packet types that they require and then waiting for the server to stream new packets to them. In this manner, clients relinquish control to the server and only get called when new data has been received for them. Clients can register for either numeric or waveform packets. When requesting numeric data, Propaq packets from the monitor are routed to the requesting clients if they contain updates to numeric values. Similarly, auto waveform packets (i.e. EKG waveform) from the monitor will be routed to clients registered for waveform packet types. Figure 9.3-3 shows the flow diagram for the client/Propaq server push protocol.

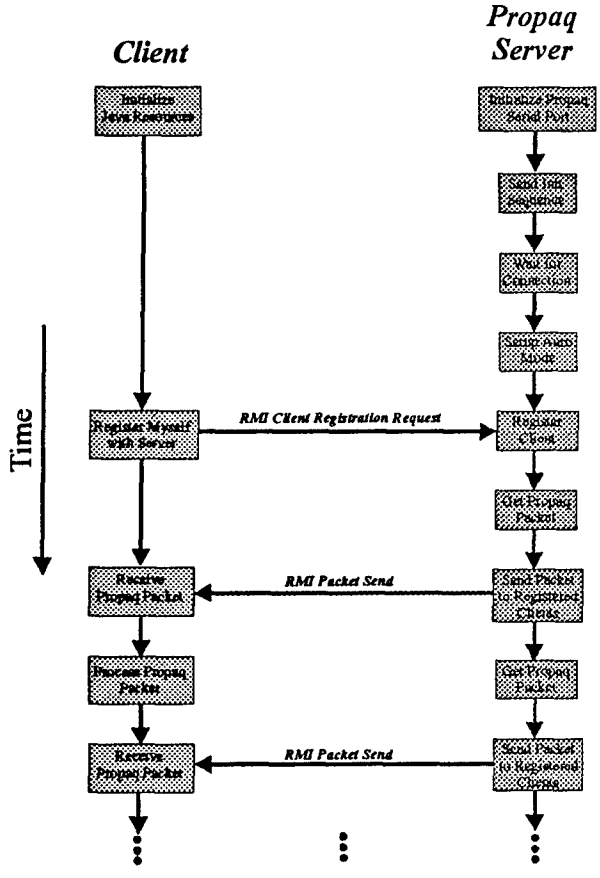


Figure 9.3-3. Propaq Server communications protocol.

Once the server has established a connection to the monitor, clients requiring data from the monitor client register this request. This will register the client with the server by adding the client reference to an internal server linked list along with any other clients that have already registered previously. When a packet is received from the monitor, the server will extract the packet from the serial port and perform a traversal of the internal linked list. If a node on the list contains a reference to a client that required the type of packet received, the server will execute a client callback function on the client application with the packet as the input parameter. The client can then process the packet in its callback function. All clients requiring packets from the server have to implement a *PropaqCallbackInterface* in the class that will process the packets. This interface contains the definition of the *PropaqAction* function that will be called by the server during the callback execution.

9.3.1 Propaq Interface

The API for the Propaq Server is implemented by the Propaq Interface as given in Listing 9.3.1-1 and consists of the calls required for client to register with the server for the push protocol, as well as methods for interactive query of the monitor.

```
package dreams.propaqserver;
import java.rmi.Remote;
import java.rmi.RemoteException;
public interface PropaqInterface extends Remote {
        public void addPropaqListener(PropaqCallBackInterface listener, byte packetType ) throws
                RemoteException;
        public void removePropaqListener(PropaqCallBackInterface listener)throws RemoteException;
        public void sendPacket( byte[] buffer ) throws PropaqException, RemoteException;
        public void sendPacket( PropaqPacket packet ) throws PropaqException, RemoteException;
        public void startAutoMode() throws PropaqException, RemoteException;
        public void stopAutoMode() throws PropaqException, RemoteException;
        public void updateBP() throws RemoteException;
        public void setBPInterval( int interval ) throws RemoteException;
        public void startSimulation()throws RemoteException;
        public void stopSimulation()throws RemoteException;
        public void pauseSimulation()throws RemoteException;
        public void unPauseSimulation()throws RemoteException;
        public boolean isSimulationOn()throws RemoteException;
        public boolean isSimulationPaused()throws RemoteException;
```

Listing 9.3.1-1. Propaq Server interface definition.

Clients register with the server by calling the *addPropaqListener* function on the server side and identify themselves as the *PropaqCallbackInterface*. The *deletePropaqListener* call is included when a client does not require Propaq data any longer. Clients can also send packets directly to the monitor using the *sendPacket* functions for accessing the monitor directly.

In order to support demo mode, a set of simulation functions are included for setting the server in simulation (demo) mode. In this mode, the server will generate packets directly to the clients with no connection to the monitor with a ten-minute loop interval.

9.3.2 Propaq Server Dependency Diagram

Figure A-7 (Appendix A) shows the dependency diagram for the current implementation of the Propaq Server java classes. The main class is found in the *PropaqServer.class* file in the *dreams.propaqserver* package. This class depends on a set of 15 other supporting classes for implementing the required server functionality.

The Propaq skeleton class is generated automatically by the RMI compiler and implements the communication infrastructure for supporting distributed communications to remote clients. Packets received from the monitor use the *CRCObject* class to check or generate valid CRC codes for the packet error correction section. Packets received from the monitor are buffered by the *PacketBuffer* class for post-processing. Propaq packet IDs for packet types and command values are finalized values in the *PropaqID* and *PropaqCommand* classes. These values are constants and predefined by the monitor communication protocol.

When a packet has been received from the monitor, an event is generated by the *PropaqEvent* class that causes the server to process the packet. The *PropaqEventListerner* will execute a *PropaqAction* function whenever the new packet is received. If an exception occurs during execution of the server, a *PropaqException* class is provided for generating custom errors for exception handling. Registered clients in the server are maintained in a linked list implemented by the *PropaqEventMulticaster* class. This class implements the methods for sending packets to all the registered clients in the list.

9.4 Run Record Server

The Run Record Server is executed on the Physician Station and is used for transferring run record data files to the Physician Station Client for display to the attending physician. Figure 9.4-1 shows the different components that comprise the current implementation of the server.

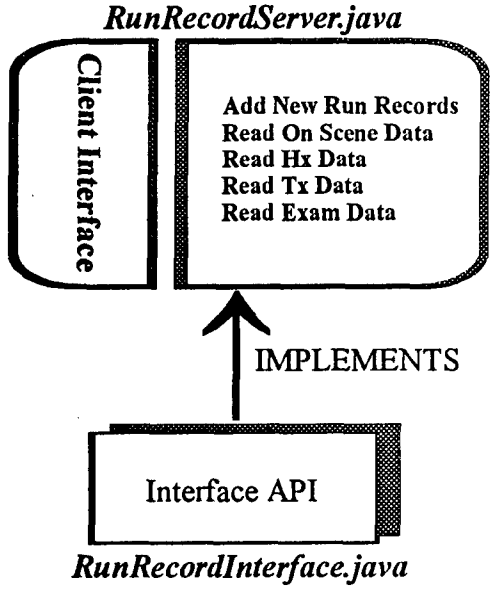


Figure 9.4-1. Run Record Interface components.

The server is composed of two main files for implementing the server functions and the interface required by the clients using the server. The functions in the server include:

- Adding new run records.
- Reading On Scene Panels.
- Reading History Panels.
- Reading Treatment Panels.
- Reading Exam Panels.

These functions are used by the Physician Station client to read updates to the currently displayed emergency records. The Paramedic-Mobile Station client maintains a set of data files with all the currently active records in the system. Each file representing the particular panel and patient identification of that record. When the emergency personnel input updates, the client will write the new record information to the local file system. The run record will then pick up the new data files and transfer the information to the Physician Station for an update. Figure 9.4-2 shows how this system is implemented.

Communications to the Run Record Server is done only by the Physician Station to update the local run record information. Listing 9.4-1 shows the functions available to clients for interfacing with the server.

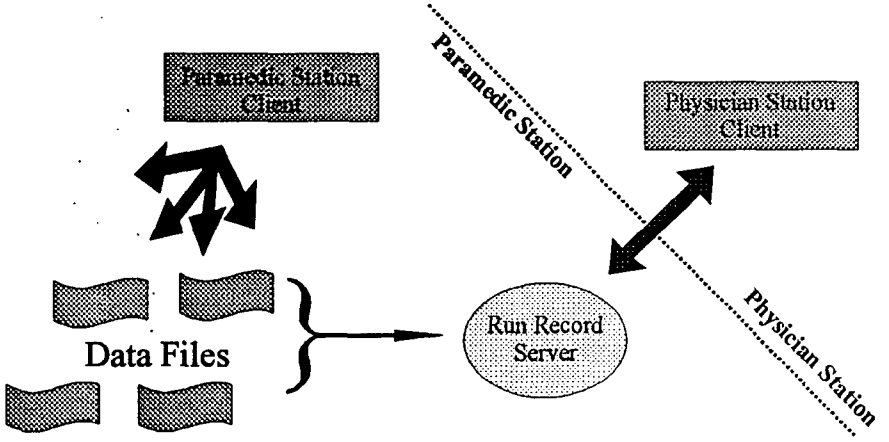


Figure 9.4-2. Run Record data transfer to physician client.

```
package dreams.runrecordserver;
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface RunRecordInterface extends Remote {

        public boolean[][] findFile()throws RemoteException;
        public boolean readAndRev() throws RemoteException;
        public String[] onScreenReadDataFromDisk (String existingFileName)throws RemoteException;
        public String[] examReadDataFromDisk (String existingFileName)throws RemoteException;
        public String[] txReadDataFromDisk (String existingFileName)throws RemoteException;
        public String[] txReadDataFromDisk (String existingFileName)throws RemoteException;
        public String narReadDataFromDisk (String existingFileName)throws RemoteException;
```

Listing 9.4-1. Run Record interface definition.

The main function of the server is implemented by the *RunRecordServer.class* file. This file implements all the functions defined by the run record interface. Figure A-8 (Appendix A) shows the dependency diagram for this server.

9.5 Sound Server

This server interfaces with the sound card hardware for voice recording of the emergency personnel in the vehicle or attending physician at the hospital site using a server pull protocol. Clients interface with the remote server by extracting sound packets from the server audio hardware and playing the sound on the local machine. Sound is recorded using a set of native interfaces that use the Microsoft Direct Sound API to access the local hardware. Figure 9.5-1 shows the different components of the current Sound Server implementation.

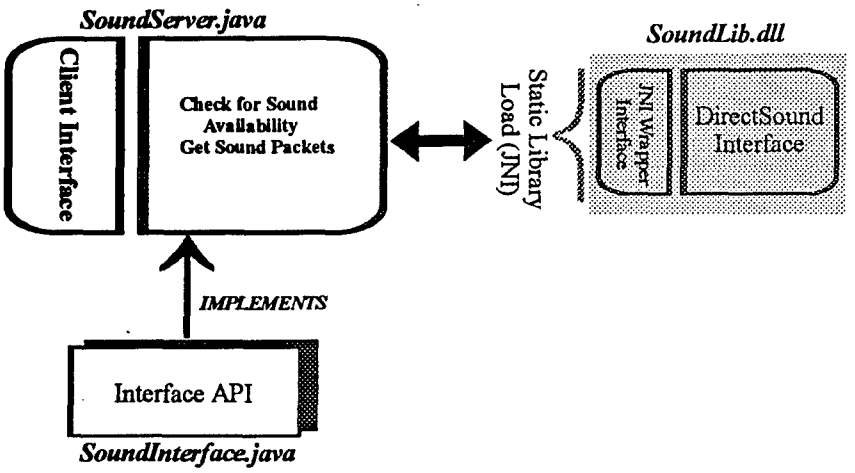


Figure 9.5-1. Sound server components.

The server consists of a single main function implemented in the *SoundServer.class* file that uses a *SoundLib.dll* for accessing the local hardware. The Direct Sound provides an API for initializing the local audio hardware, querying the hardware for available audio data, and extracting recorded data streams when audio data is available. Listing 9.5-1 shows the interface implementation for the current version of the server.

```
package dreams.soundserver;
import java.rmi.Remote;
import java.rmi.RemoteException;
public interface SoundInterface extends Remote {
        public boolean soundClipAvailable() throws RemoteException;
        public byte[] getSoundClip() throws RemoteException;
```

Listing 9.5-1. Sound Server interface definition.

The *soundClipAvailable* function is used by clients for checking the server for any new audio data. A separate thread is kept by the server which polls the local hardware for new audio data every 500 msecs. When data is available, the *dataAvailable* flag is set in the server and clients can then call the *getSoundClip* to receive the recorded data. Clients in the system always remain in control during communications and only ask for audio data when the local sound buffer has been played and new data is required. Current implementation of the server limits the number of clients that can connect to it; therefore, when implementing a full duplex communications link between vehicle personnel and the attending physician, two pairs of client/server applications are executed at each end of the link. A sound server will execute on the hospital side, which streams audio data to the client system. Similarly, a sound server is also executed on the Paramedic-Mobile Station for streaming audio data to a client at the hospital side.

Figure A-9 (Appendix A) shows the dependency diagram for the main Sound Server class.

Only two Java class files are required for supporting the main *SoundServer.class* in the *dreams.soundserver* package. The SoundServerInterface.class defines the two call available for the clients in the system. A skeleton class is also required as with all other servers for implementing the communication between the client and server. The native interface for accessing the Direct Sound functions is implemented in the *SoundLib.dll* function. This library will use the *Dsound.dll* library which implements the Direct Sound calls defined by the API.

9.6 GPS Server

The GPS Server provides the location and time to registered clients in the system at a predefined update rate. Currently, the GPS Server is limited to providing a set of simulation locations for use in the mapping server computations. Further work will include integration into a GPS receiver for providing actual location points. Figure 9.6-1 shows the current server components for the GPS server.

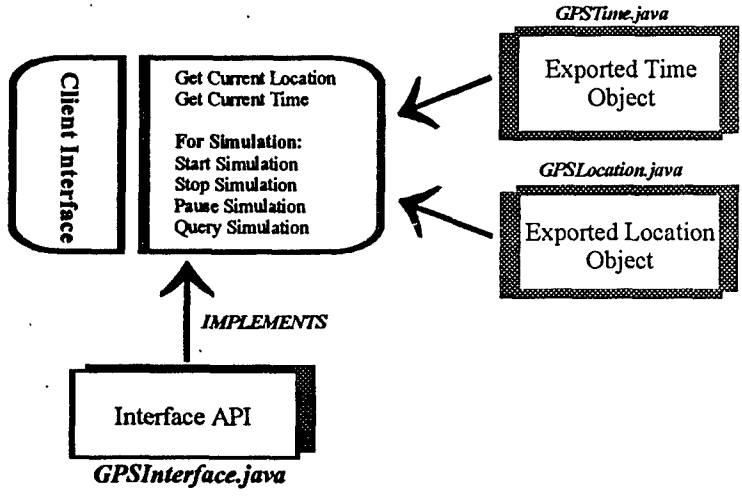


Figure 9.6-1. GPS server components in system.

Current GPS server implementation depends on two classes that define a Java location object and a Java time object. These objects are used to transfer the locations and time values to requesting clients that require current absolute time and/or vehicle locations.

Client functionality is supported by the GPS Interface that defines the functions implemented by the server. Listing 9.6-1 shows the current interface available to clients.

```
package dreams.gpsserver;
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface GPSInterface extends Remote {
        public GPSLocation getGPSLocation() throws RemoteException;
        public GPSTime getGPSTime() throws RemoteException;
        public void resetGPSServer() throws RemoteException;
        public void startSimulation() throws RemoteException;
        public void stopSimulation() throws RemoteException;
        public void pauseSimulation() throws RemoteException;
        public void unPauseSimulation() throws RemoteException;
        public boolean isSimulationOn() throws RemoteException;
        public boolean isSimulationPaused() throws RemoteException;
```

Listing 9.6-1. GPS Server interface definition.

This interface supports a set of simulation functions for generating simulated location values when the system is in demo mode. A simulation client will request a start simulation command to the server to switch from location acquisition to demo location use. Further requests for locations will be given a simulated location.

Server implementation constitutes a set of five class files that implement the interface functions and define the location and time objects. Figure A-9 (Appendix A) shows the dependency diagram for the main Java GPS server class.

The main server class is defined by the *GPSServer.class* function which implements the *GPSInterface.class* API functions. A skeleton function implements the communications infrastructure for client/server interactions using RMI. The last two files (*GPSLocation.class* and *GPSTime.class*) define the location and time objects returned by the server to requesting clients.

9.7 Map Server

The Paramedic-Mobile Station and Physician Station both have a navigation system that uses a Mapping Server for extracting mapping information and to map display images. This server uses a separate mapping application to generate map images for returning to clients, and tracks the current position of the ambulance. The current location is obtained from a GPS Server executing within the local subnets in the vehicle. This location is then used for generating a tracking location on the current map.

The overall Digital EMS navigation system is comprised of the Map Server, GPS Server, a client on the Paramedic-Mobile Station, and a client on the Physician Station. The Map Server implements all of the methods that need to use the Streets32 software package. It controls the software by issuing commands and returns the resulting map to the client. The GPS Server supplies the current location coordinates every few seconds. The clients display this location on the map provided by the map server on the interface. As mentioned earlier, the GPS server currently simulates coordinates along a route. When a GPS is integrated with the hardware on board the ambulance, the GPS server will obtain the location and provide it to the clients. The servers and clients communicate via Java RMI.

Current implementation of the map server consists of a set of class files that interface with the Streets32 application using a set of script commands which are read in by the Streets32 real-time monitor. Script commands instruct the mapping software to generate map image files that are then converted to a JPEG format and routed to the requesting client. Figure 9.7-1 shows the current components which make up the server software.

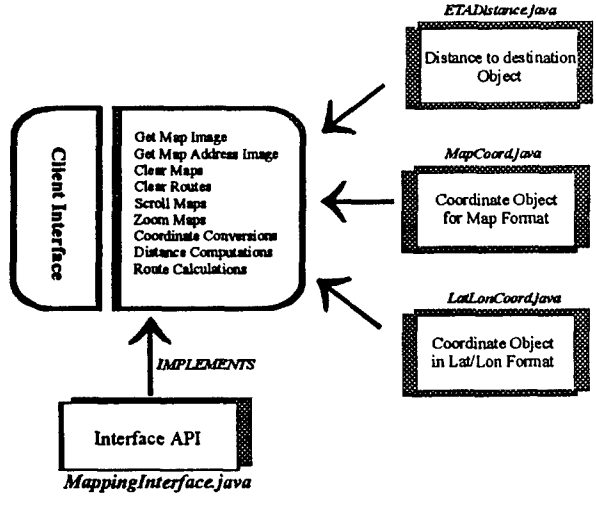


Figure 9.7-1. Map Server components in system.

The Map Server main class implements a set of functions for map manipulation and route finding through the use of the *Streets32* software. A set of supporting class files (*MapCoord.class* and *LatLonCoord.class*) is used for representing the different coordinate systems used by the software to pinpoint map locations. The interface file is used for interfacing with the clients in the system and is shown in Listing 9.7-1.

```
package dreams.mapserver;
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.awt.Image;
import javax.swing.ImageIcon;
import java.awt.*;
public interface MappingInterface extends Remote {
        public ImageIcon getMap(String AmbulanceID, boolean center_mode, int x, int y ) throws
                RemoteException;
        public ImageIcon getMapRouteAddress(String location1, String location2) throws
                RemoteException;
        public ImageIcon clearMap() throws RemoteException;
        public ImageIcon scrollMap(int direction) throws RemoteException;
        public ImageIcon getZoomedMap(int x, int y, float zoomVal ) throws RemoteException;
        public ImageIcon getZoomedMapMulti(String ambID1, String ambID2,int x1, int y1,
                int x2, int y2, int choice, float range) throws RemoteException;
        public String getDistanceETA(String location,int x, int y) throws RemoteException;
        public String XYtoLatLon(int x, int y) throws RemoteException;
        public void clearRoute(String AmbulanceID,int x, int y) throws RemoteException;
        public Point getCoordFromAddress(String address, String zip) throws RemoteException;
        public Point getCoordFromCross2St(String address1, String address2) throws
                RemoteException;
        public Point getCoordFromLatLon(float lat, float lon) throws RemoteException;
        public Point getCoordFromPoint(String point) throws RemoteException;
```

Listing 9.7-1. Map Server interface definition.

9.7.1 Streets32 Development

The mapping software chosen for the navigation system is Streets32, developed by Klynas Engineering (www.klynas.com) for Windows platforms (screen shot in Figure 9.7.1-1). It includes a database of maps including many levels of streets, boundaries, and railroad tracks. On most city maps, schools, churches, hospitals, and other landmarks are also included. It provides a script-based programming interface through which other applications can execute map processing commands. A GPS interface is available that can be used to display moving vehicles on the map. Any number of vehicles can be tracked and displayed with a unique token and name. It features a routing system that can provide travel directions from source to destination using landmark names, cross streets, X-Y or latitude-longitude coordinates. The route can be calculated based on a number of factors including average speed of a street, travel cost per mile, fuel usage, etc. The route can be displayed on the map, and a detailed travel report is created, which can be used to determine the estimated time for the route. It also provides an extensive search capability based on street name, address, cross street, zip code etc. The map can be saved to the clipboard or to an EMF graphics file for use by other applications in the system.

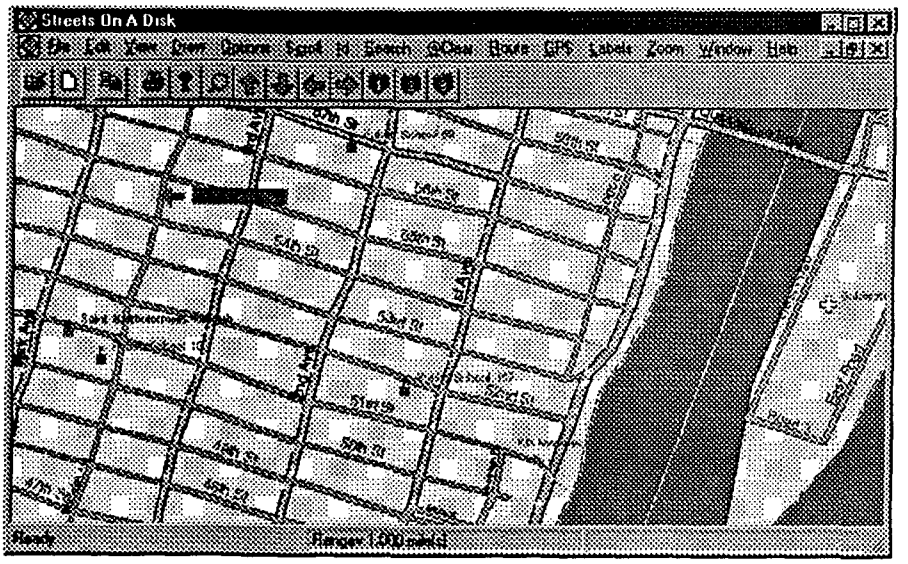


Figure 9.7.1-1. Streets 32 mapping software.

The map server controls the Streets32 software depending on the clients' request. It implements the functions that:

- Provide a map with the ambulance displayed at the current location.
- Provide a map with the travel route to the destination.
- Provide a zoomed map.
- Provide a map scrolled in a specified direction.
- Clear the calculated route on the map,

Additionally, the map server also has functions needed to calculate the route and obtain the estimated time of travel from the travel report, etc.

9.7.2 Map Server Dependency Diagram

Figure A-10 (Appendix A) shows the dependency diagram for the main Java Map Server class.

9.7.3 Additional Mapping & GPS Development

GPS and Differential GPS (DGPS) Receivers. The GPS satellites broadcast at two base frequencies, which are referred as L1 and L2 bands (L1=1.575GHz and L2=1.228GHz). Two pseudo-random noise codes are modulated onto these two base carriers. This spread spectrum technology is applied to avoid intentional or unintentional jamming. The first code is the C/A-code (Coarse/Acquisition-code), which is available for civilian use. The C/A-code is modulated on L1 band only. The second code is the P-code (Precision-code), which has been reserved for U.S. military use. Without SA (Selective Availability), the accuracy of C/A-code positioning is in the range of 30 m. The atmospheric distortion and clock errors are two major sources of error. For security reason, SA is imposed by the U.S. Department of Defense to

achieve intentional degradation of accuracy for civilian users. With SA, the accuracy is in the range around 100 m. Differential GPS (DGPS) was later developed to overcome the full range of errors including the inaccuracies of Selective Availability. With DGPS, position accuracy can be achieved to within two to five meters. Details about DGPS will be added in the next section

For a GPS receiver to locate a satellite and then receive the satellite signal, it must set a channel to look for a particular satellite. There are basically three situations: hot start, warm start, and cold start. The differences among these three situations are how long the receiver has been powered off, if it has any information stored, and how accurate the information is.

DGPS uses a reference station to determine the errors in the satellite signal and generate correction data. The correction data is then applied to refine the position. The correction can be done at GPS receiver side or reference station side. In the first case, the correction data is broadcast or transmitted through some dedicated channel to the GPS receiver, which should be DGPS correction capable. The U.S. Coast Guard Maritime DGPS Service broadcasts correction signals on maritime radio-beacon frequencies. Some private companies, such as Differential Correction, Inc., provide DGPS service by broadcasting correction signals on FM radio. The standard format for transmitting correction data is called RTCM SC-104. The second case is called inverse DGPS, in which the satellite signals received by the GPS receiver are transmitted to the base station and then corrected. If the receiver needs an accurate position, the corrected position data are then feed back. Notice that inverse DGPS requires building a DGPS base station.

Digital EMS requires that a level of repeatability and relative accuracy consistent with DGPS be provided to the EMS crew. There are many cases where GPS positions have been inaccurate enough to place ambulances at the wrong house and even on the wrong side of intersections. Therefore, the DGPS accuracy will only benefit the EMS crew when coupled with mapping and navigation software. It may also benefit the physician somewhat since an estimate of how long the patient will be in transport may be helpful when the physician decides to take certain approaches. With a frequently updated and well-maintained database of roads and streets, it is not difficult to predict how effective and helpful the GPS/mapping system would be to the EMS driver. Since this project is basically in development, DGPS should be included for further evaluation.

Building a DGPS base station is definitely more cost-effective in the long term, particularly when many ambulances are deployed. That eliminates the need to receive DGPS correction signals on each ambulance. However, there are two drawbacks of this approach. It costs approximately $7,000 to set up a base station, not including the development cost. (A good DGPS station receiver with a surveying grade antenna from Ashtech or NovAtel costs approximately $4,000 to $4,500. The inverse DGPS processing software from Waypnt, a Canadian consulting company, costs $2,000. A computer is needed to host the software.) A DGPS beacon receiver costs only $750 (service is free), while an FM receiver with service costs averages approximately $600 per year. Both are significantly more cost-effective than a DGPS base station. Though it does not consume much communication bandwidth, it considerably complicates the communication management task. Because the benefit of a DGPS in a digital ambulance is not yet known, a base station has not yet been built. Both College Station and Houston are covered by the U.S. Coast Guard DGPS broadcast site on the Galveston Island. A DGPS base station is currently available to Digital EMS through the Houston Police Department.

Automatic Vehicle Location System. Automatic Vehicle Location (AVL) is a technology used for tracking vehicles, vessels, and mobile assets such as trailers, containers, and equipment. Each mobile unit has a GPS receiver that reports its position to the base station over a communication network. This allows the base station to monitor the entire fleet and manage the mobile assets. AVL makes dispatch much

faster, especially when integrated with Computer Aided Dispatch systems (CAD). That makes AVL extremely attractive to those who provide emergency services, such as hospitals, fire departments, and police departments. There are some companies developing AVL systems, such as Trimble Navigation Limited and Baker Geo Research, Inc. However, all commercial products are focused on vehicle location and fleet management from the viewpoint of the base station. None of them provides direct and in-vehicle mapping/navigation capability to the vehicle crew, which is exactly what is proposed to be integrated with Digital EMS. Moreover, a major component of those commercial products is the wireless communication network and the associated communication management software. It would be less cost-effective to buy any AVL system products since our project has already implemented a communication network and Intelligent Communication Manager.

GPS Systems Market Survey. Three characteristics are critical for choosing a GPS receiver. First, the GPS board or unit must have at least two serial RS-232 ports, one for position reporting and the other for DGPS correction data. Secondly, it must be DGPS-ready. Finally, it should have a sufficient number of receiver channels. For the vehicle tracking application, it would be advisable to have eight channels. A twelve-channel system would be even better. After an extensive search, three companies have been located that provide receivers that meet the Digital EMS requirements: Canadian Marconi Company (CMC), Ashtech, and Trimble Navigation. Each has two products suitable to the Digital EMS application. Figure 9.7.3-1 lists the characteristics of six OEM GPS boards manufactured by them.

GPS SYSTEMS SURVEY						
	AllStar	SupterStar	G8	G12	Lassen-SK8	Lassen-SKII
Manufacturer	CMC	CMC	Ashtech	Ashtech	Trimble	Trimble
# of channels	12	12	8	12	8	8
Development kits available	yes	Yes	yes	Yes	yes	yes
RS-232 Serial ports	2 TTL levels, one duplex one half-duplex for RTCM		Same as AllStar	two full duplex	dual	dual full duplex
Power	10-16V	10-16V	5V(+5%)	5V(+5%)	5V(+5%)	5V(+5%)
Hot Start		15 sec	10 sec	11 sec		20 sec
Warm Start		45 sec	45 sec	35 sec		45 sec
Cold Start		2 min	2 min	2 min		130 sec
Reacquisition		1 sec	1 sec	2 sec		2 sec
RTCM support		1, 2, 9	1,3,6,9,16	1,2,3,6,9,16		
Accuracy						
DGPS		<1m	3 m	~ 0.5 m		2 m
Without SA		<16m				24 m
with SA		<40m	40m			
DGPS capable	yes	Yes	yes	Yes	yes	yes
DGPS base?	yes	No	no	Yes	no	no
price with kits	$635	$650	$750	$2,495	~$1,000	~$1,000
Board only			$140			

Figure 9.7.3-1. GPS systems survey.

Future Mapping System Requirements and Software Development. The following list presents requirements suggested for future map software development.

- Continually read GPS positioning data and update the vehicle's position on the map.
- Calculate dead reckoning based on GPS coordinates.
- Show the coordinates of any point specified by a mouse click, which is called geocoding.
- Searching a place on the map by street address and/or ZIP code.
- Finding the nearest street to a GPS position (longitude/latitude), which is called reverse geocoding.
- Compatibility with other map formats.
- Provide choice of map projection
- Provide choice of map datum
- Provide choice of map scale
- Display maps with zooming, panning, and scrolling functions.
- Calculate routes and estimate travel time.
- Add point and shape objects on the map.
- Report the route and estimated travel time to base station once a route is chosen.
- Show the direction of heading.
- Rotate the map to track heading direction (as optional.)
- Provide a North up view for comparison to printed maps

After a thorough search of mapping development software tools, four candidates have been found. All of them are Microsoft Windows-based products. Based on the above requirements, MapObjects 2.0 from ESRI is the most promising. MapObjects is an ActiveX control (OCX) with more than 45 programmable ActiveX automation objects. Since our system is built in platform independent Java architecture, the challenge is then shifted to integration of ActiveX controls with our existing Java software system. Figure 9.7.3-2 shows the different mapping software development environment that meet the Digital EMS requirements.

MAP DEVELOPMENT SOFTWARE				
Product	MapObjects 2.0	MapX 4.0	MapOCX Pro	GeoEngine
Company	ESRI	MapInfo	Chicago Map	Etak
OS System	Win95/98, NT	Win95/98, NT	Win95/98, NT	Win95, NT
Development Environment	VB, VC++, Delphi, Powerbuilder (PB)	VB, VC++, Delphi, PB, Lotus Notes	VB and Delphi	VB and VC++
ActiveX Control	Yes	Yes	Yes	DLL based
Route Finding	Yes with a downloadable DLL	Yes	Yes	Yes with Path-finding plug-in
GPS Input	GPS tracking	GPS tracking, read long./lat.	Read long./lat. in NMEA format	Ready Using serial port
Coordinate Systems	More than 1 support projection	Support all Projections	Just 1 Unprojected	Support projection
# of components	>45	31(many methods)	~150	>100 DLL's
Map Data Set	Format matching	Mapinfo tab type	Chicago Map Supply only	Etak map data set only
Geocoding	Add. Street level default at ZIP	ZIP level, street Level: MapMarker	No geocoding	

MAP DEVELOPMENT SOFTWARE				
Inverse Geocoding	6 digits	6 digits	6 digits	5 digits
Support	Prompt, efficient, developer forum online documents	Good documents Discussion center	Manual only	Seems little
Scroll	Yes			
Price	$5,000 with street level map data, 40 distribution seats, $1,000/developer	$15,000 including Mapmarker and Texas street level Maps	$429 add $40 per distribution seat	$15,000 standard tool kit plus Pathfinding plug-in and data set
Comments	No. 1 in GIS SW Based in CA	Based in Troy, NY Public, worldwide	Small company	Belonging to The Sony group

Figure 9.7.3-2. Map development software.

10 System Hardware Architecture

Connectivity for the emergency vehicle systems includes the following components:

Figures 10-1 and 10-2 show the system architecture and interconnection strategy for the current implementation of the Digital EMS system.

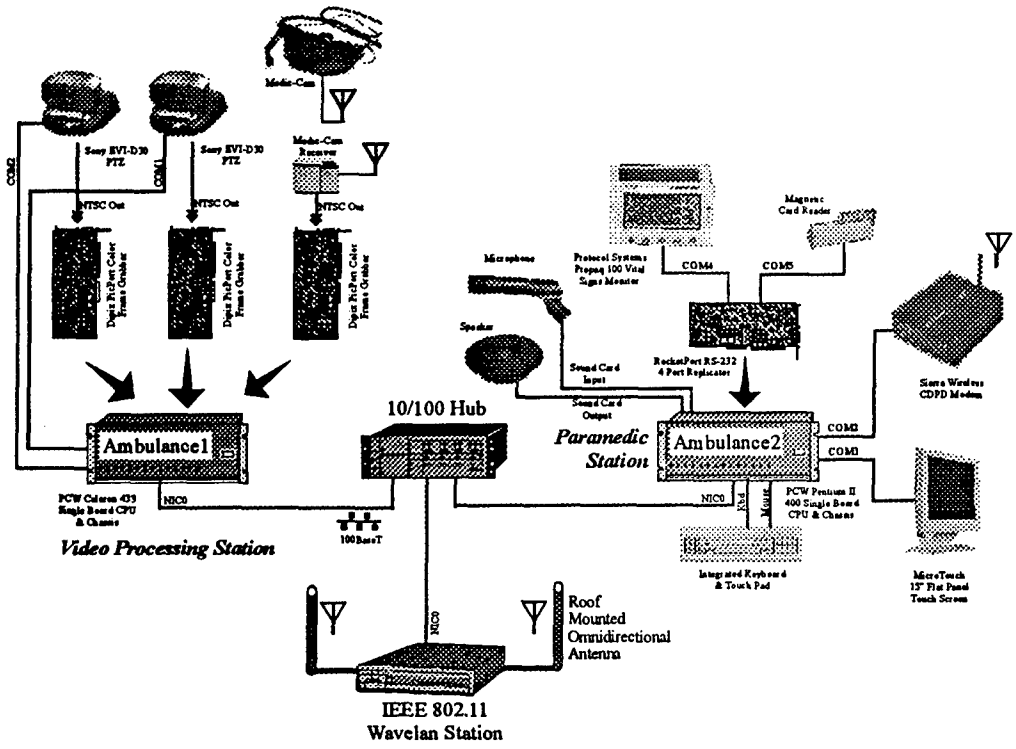


Figure 10-1. Hardware setup and connectivity for emergency vehicle systems.

- Ambulance1 (Video Processing System):

- Dipix Picport Color Frame Grabbers (3).
- Sony EVI-D30 PTZ cameras (2).
- Army's Medic-Cam Receiver.
- RS-232 Camera Control Inputs (2).
- 100BaseT Ethernet connection.

- Ambulance 2 (Paramedic-Mobile Station System):
  - Microtouch flat panel video input.
  - Microtouch flat panel touch input – RS-232.
  - Integrated keyboard/touch pad.
  - Rocketport (4 port) RS-232 port replicator.
  - Card reader connection.
  - Propaq connection.
  - CDPD input connection – RS-232.
  - Sound Blaster sound card.
  - Microphone.
  - Speaker.
  - 100BaseT Ethernet connection.
- 10/100 BaseT Hub:
  - Ambulance 1 Ethernet.
  - Ambulance 2 Ethernet.
  - Breeze 802.11 Wireless Lan hub.

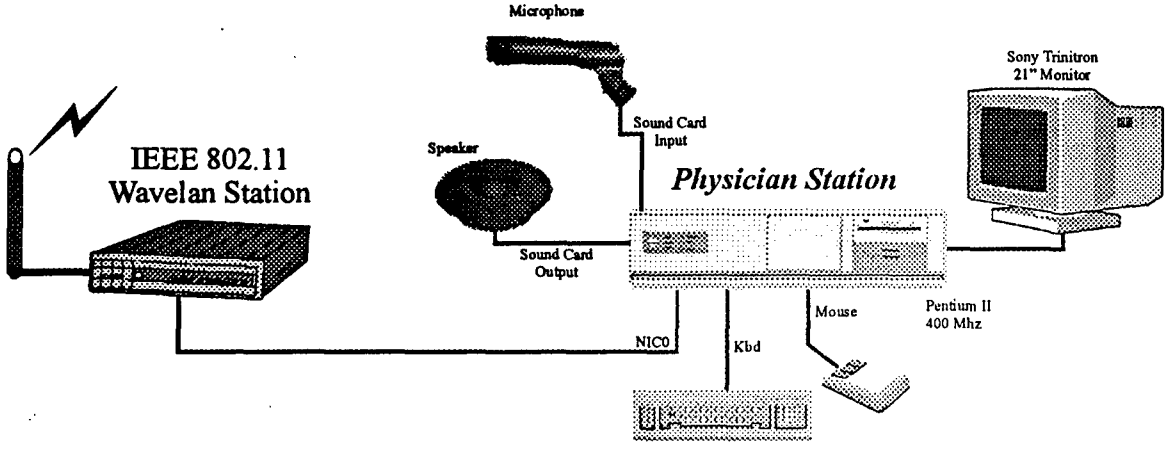


Figure 10-2. Hardware setup and connectivity for the hospital side workstation.

Connectivity for the hospital side system includes the following components:

- Physician Station:
  - 10/100 BaseT Ethernet network connection
  - 100BaseT Breeze 802.11 Ethernet on local or mounted segments.
  - Microphone.
  - Speaker.
  - Keyboard.
  - Mouse.
  - 21" Display.

Figure 10-3 shows a cut-away diagram of the ambulance with the mounted flat panel display and integrated keyboard/mouse sitting on the keyboard notch by the operator's seat. Vehicle systems are located in the radio compartment below the keyboard. Access to these systems is only available from the outside of the vehicle.

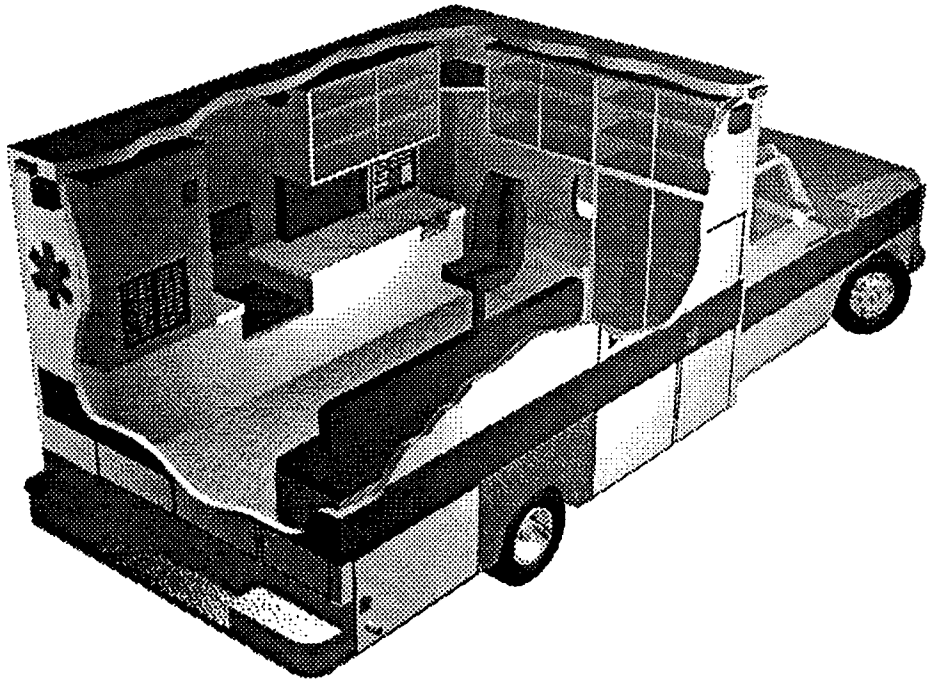


Figure 10-3. Emergency vehicle configuration.

11 User Interface Description and Overview

The Digital EMS system architecture currently facilitates simultaneous interaction between two separate workstations: a Paramedic-Mobile Station on board the Digital EMS ambulance and a Physician Station located at the hospital. This section contains an explanation of every interface screen from both the Paramedic-Mobile and Physician Stations, as well as an explanation of each screen's usage and purpose. Screen shots of each interface screen described in this section can be found in Appendix B.

The EMS personnel on board the ambulance can benefit greatly from technology integration. Several time-saving technologies include a driver's license scanner, a touch-screen display, an interactive protocols manual and, in the near future, a medication barcode scanner. Java RMI technology enables communication of vital patient data, including a Propaq vitals signs monitor (Protocol Systems, Inc.), three VISCA/RS-232C compliant computer controllable cameras, video capture cards, and a host of wireless communication products. These technologies have been integrated into the Digital EMS system to not only expedite mobile patient care but also to extend the realm of treatment options by providing the hospital physician a window into the transiting ambulance.

This section will begin with the interfaces and capabilities available to the hospital physician and then cover those available to the ambulance personnel. Data entry forms in the Digital EMS system contain various interface objects to expedite data entry. Text fields allow a single line of text from the keyboard, whereas text areas are used to allow multiple lines of text. Combo boxes, also known as combination boxes, allow users to select from a drop-down list of values or type in their own custom response. Option boxes enable the user to select one and only one within a group, whereas checkboxes can be selected/deselected as the user desires. Other custom interface objects, such as buttons, will be described below with reference to their respective interface screen in Appendix B.

11.1 Physician Side Interfaces

The Physician Station at the hospital consists of two primary displays, as demonstrated in Figure B-1 (Appendix B), one focused on video and the other on data. The physician can switch between these two displays at any time or, in the near future, have each displayed simultaneously on separate monitors. The Video display consists of an "**inter*act***" video window, a row of thumbnail camera buttons along the top of the display, and a side panel of camera controls. The Data display consists of a main window with a raised system button panel located across the bottom of the screen. Pushing one of the three system function buttons will cause the contents of the window to change. The Digital EMS interface is designed such that the system button panel will be displayed at all times that the Data Display is active. The physician Video and Data Displays will be discussed below in more detail.

A key component of the Digital EMS system is the "behind the scenes" data communication between the ambulance and the hospital physician. Patient vital sign data is transmitted real-time from the ambulance to the physician. Patient run record data is transmitted from the ambulance to the physician at frequent, automatic time intervals. Video of the patient is transmitted from the ambulance to the physician at a frame rate and quality relative to the available bandwidth. Camera rotation and zoom commands are transmitted real-time from the physician to the active ambulance camera. The continuous communication and interaction between the physician and the ambulance system provides a "window" of opportunity to advance mobile patient care.

11.1.1 Physician Camera Interface

Upon switching to the Video Display, the physician can immediately view video of the patient as well as control the cameras on board the ambulance. Through the integration of three VISCA/RS-232C compliant cameras and a video capture system, the Digital EMS system equips the hospital physician with tools necessary to intervene early in the care of trauma patients.

The interface design includes a main "**inter*act***" video window, an Active Camera thumbnail panel, and Camera Control side panel, as shown in Figure B-2 (Appendix B). The physician can switch between the three on-board cameras by pressing the thumbnail buttons across the top of the screen, with the active camera thumbnail highlighted in red. All three thumbnail images are automatically updated at a specified time interval, allowing the physician multiple view ports. A U.S. Army Medic-CAM head-mounted display and camera has been integrated into the Digital EMS system, allowing the physician to view video of EMS personnel outside the ambulance. The physician can select the Medic-CAM as a video source by pressing the "Medic-CAM" button. The Active Camera display also includes a "Setup Local Video" button that enables the physician to adjust his/her camera.

The Camera Control panel enables the physician to manipulate the cameras on board the ambulance. The top set of buttons control the rotation of the camera, with the active camera number displayed in red in the center. Upon pressing one of the rotation buttons, a control command will be sent to the active remote camera causing it to move in the specified direction. The buttons can either be tapped for minor adjustments or held pressed for continuous panning. The Zoom Control buttons can be used in a similar fashion to zoom closer to or farther away from the patient. In addition, this panel includes a Presets window that enables the physician to quickly rotate and zoom the active camera to one of four preset locations: the patient's head, chest, groin area, and ankles. The physician can either press a preset box to activate that preset view or press one of the three camera icons to activate that camera. The icon of the active camera is highlighted in red.

11.1.2 Physician Vital Signs Interface

Upon switching to the Data Display and pressing the "Vitals" button at the bottom of the screen, the physician can view the patient's vital signs in real-time, as demonstrated in Figure B-3 (Appendix B). This is made possible through the integration of a Propaq Vitals Signs Monitor into the Digital EMS system. Since the patient's vital signs are continuously transmitted to the hospital, the physician on call will always have current information on the patient's condition. In the near future, this "Vitals" screen will also allow the physician to command the Propaq to immediately take a blood pressure reading by pressing the "Update BP" button as well as set an automatic timed interval using the "Set Interval" button.

11.1.3 Physician Map and Navigation Interface

On the Physician Station, multiple ambulances can be tracked at any time. In addition to displaying the map, the navigation system has other capabilities including zooming in and out and scrolling in all directions. The current coordinates of the ambulance, estimated time of arrival at the hospital, and the distance from the hospital in miles are also displayed. A pull-down menu with a list of ambulances lets the physician select an ambulance, and control switches to that particular ambulance. A screen shot of the Physician's Station map interface is shown in the Figure B-4 (Appendix B)

11.1.4 Physician Run Record Copy

Upon pressing the "Run Record" button at the bottom of the Data Display screen, the physician will be able to view patient run records, as shown in Figure B-5 (Appendix B). The run record is subdivided into five pages of forms with the first page, entitled "On-Scene," containing basic patient and incident information and a motor/vehicle survey. All run record data entered by ambulance personnel is automatically transmitted to the hospital at specified time intervals, thus ensuring that the physician is viewing up-to-date run record data at all times.

Located in the bottom border of the "**inter**act" window are a small arrow button and read-only field identifying the patient's name. The physician can switch between patients' forms by pressing the small arrow button, which displays a pop-up list of patients currently stored in the system. The physician is restricted to read-only viewing of the run records and cannot modify any data. For further details about this form, please see the Paramedic-Mobile Station "Run Record" section.

The "Exam/Observation" page of the run record contains the primary survey of the patient, as shown in Figure B-6 (Appendix B). All run record data entered by ambulance personnel is automatically

transmitted to the hospital at specified time intervals, thus ensuring that the physician is viewing up-to-date data at all times. The physician is restricted to read-only viewing and cannot modify any data. For further details about this form, please see the Paramedic-Mobile Station Run Record explanation earlier in this section.

The Patient HX page of the run record contains background history of the patient, as shown in Figure B-7 (Appendix B). This data was obtained by the ambulance personnel through verbal interrogation of the patient or automatic retrieval from the hospital database system. All run-record data entered by the EMS personnel is automatically transmitted to the hospital at specified time intervals, thus ensuring that the physician is viewing up-to-date data at all times. The physician is restricted to read-only viewing and cannot modify any data. For further details about this form, please see the Paramedic-Mobile Station Run Record explanation in the following section.

The Patient TX page of the run record contains a timestamp log of all medications administered to the patient, as shown in Figure B-8 (Appendix B). All run record data entered by the EMS personnel is automatically transmitted to the hospital at specified time intervals, thus ensuring that the physician is viewing up-to-date data at all times. The physician is restricted to read-only viewing and cannot modify any data. For further details about this form, please see the Paramedic-Mobile Station Run Record explanation earlier in this section.

The Narrative page of the run record contains free-form text area in which the physician can view the narrative details of the incident scene, transport, treatment and treatment response of the patient, as shown in Figure B-9 (Appendix B). All run-record data entered by ambulance personnel is automatically transmitted to the hospital at specified time intervals, thus ensuring that the physician is viewing up-to-date data at all times. The physician is restricted to read-only viewing and cannot modify any data. For a further detail about this form, please see the Paramedic-Mobile Station Run Record explanation earlier in this section.

11.2 Paramedic-Mobile Station Interfaces

A sample screen of the Paramedic-Mobile Station on board the ambulance is shown in Figure B-10 (Appendix B) The screen is composed of two primary areas: a sunken "**inter**act" window outlined in a dark gray border and a raised system button panel located across the bottom of the screen. Pushing one of the seven system function buttons (see item 1 in Figure B-10) located along the window bottom will cause the contents of the **"interact"** window to change. The Digital EMS interface is designed such that the button panel and "**inter**act" window will be displayed at all times, regardless of which function the EMS personnel are currently performing.

Located in the bottom border of the "**inter**act" window is a small arrow button displaying the current patient's name at all times; this field is automatically updated when the patient's full name is entered or modified in the on-line run record. The EMS personnel can switch between patients' forms by pressing the small arrow button, which displays a pop-up list of patients currently stored in the system.

The Run Record and Report panels contain multiple pages of forms, enabled by a tab interface across the top of the window (see item 3 in Figure B-10). This sample screen contains five tabs, with the On-Scene tab currently displayed. Pushing one of the other four tabs will cause the respective tab to be highlighted and the contents of its page to be displayed. The seven system functions and sample screens will be discussed below in more detail.

11.2.1 Emergency Run Record

The on-line run record, which was originally based upon actual paper forms, is divided into a Run Record section and a Report screen. Upon pressing the "Run Record" button at the bottom of the screen, EMS personnel can view and modify time-essential patient data. The "Report" section contains more time-insensitive billing and insurance information, and is described later in this section.

It is important to note that all data entered in the run record is automatically backed up. Any time the user is switching to another run record page or to another system function, the system will automatically save the current page before switching. Also, the system utilizes a timed backup method in which the current page is written to disk at specified intervals, such as every three minutes. If the Digital EMS system were to temporarily lose power, the system run record would automatically be restored with the most recently stored data. In addition, all run record data is automatically transmitted to the hospital at specified intervals, thereby ensuring that the doctor is viewing up-to-date run record data.

The Run Record section is subdivided into five pages of forms. The first page, entitled "On-Scene," contains basic patient and incident information and a motor/vehicle survey, as shown in Figure B-11 (Appendix B). A card reader has been integrated into the Digital EMS system that significantly reduces the time required to obtain patient information. By sliding a patient's driver's license through the card reader, most of the "Patient Info" data will be filled out automatically. If the patient's driver's license cannot be obtained, this information can be manually typed in.

A capability is currently being developed to utilize patient information to automatically query the hospital database. If previous medical history for this patient is found in the database, it will be securely transmitted to the ambulance and will show up in the Patient History section of the run record. The Critical Problems section contains vital information for the physician at the hospital. Each of these items can be answered by either choosing from the drop-down list or typing in a custom response. If the EMS personnel change any of this information, it can be immediately retransmitted to the physician by pressing the "Send On-Scene Data to Hospital" button.

The Mechanism of Incident section contains a number of items as well as a vehicle figure. The vehicle can be changed by pushing the small arrow button to the right of the figure, which displays a pop-up list of vehicle types including a motorcycle, sedan, and semi. The EMS personnel can indicate the patient's seat position by pushing one of the seats in the figure, which will subsequently be highlighted in yellow. The direction of impact can be indicated by pushing the arrows surrounding the vehicle figure. Two or more arrows should be selected if multiple vehicles collided with the patient's vehicle.

The Exam/Observation page of the run record contains the primary survey of the patient, as shown in Figure B-12. Each of the survey items can be answered by choosing from the pull-down list or typing in a custom response. Note that the "Breath Sounds" and "Pupils" sections have separate left and right designations. A Glascow help button has been included to assist the EMS personnel that, when pressed, displays a pop-up window containing the various symptom values used to calculate the Glascow score.

The Type and Location section contains a human figure that is automatically determined by the data entered in the On-Scene form. Patients designated as male or female on the form will cause a male or female figure to be shown in this section, respectively. In the near future, the system will be able to display a unisex child figure if the estimated age is entered as 10 or under and an infant figure for estimated ages two years old and younger. EMS personnel can identify major problem areas by simply tapping the figure and choosing a description in the numbered list. Numbered labels can be moved by dragging them to their new location and removed by dragging them out of the figure. In the future, the

figure's head can be pressed to provide a close-up view of the patient's head, enabling precise labeling of injuries to the patient's face and skull.

The Patient HX panel of the run record contains background history of the patient, as shown in Figure B-13 (Appendix B). This data can be obtained in one of two ways: verbal interrogation of the patient or automatic retrieval from the hospital database system. The survey is ordered and based upon the SAMPLE method: Symptoms, Allergies, Medications, Past History, Last Food and Fluids, and Events leading to the incident. This page also contains a free-form Problem List in which the EMS personnel can document the serious problems and identify the approach plan.

The "Patient TX" page of the run record contains a timestamp log of all medications administered to the patient, as shown in Figure B-14 (Appendix B). A barcode scanner will be integrated into the Digital EMS system to significantly reduce the time required to log medication information. Since ambulance medication is already bar-coded for hospital billing purposes, a medicine container can be quickly scanned as it is taken from the shelf. By passing the bar code in front of the reader, the medicine name will immediately be entered into the next available row and the current time will be logged. The "Dosage" and "Route" comboboxes can even then be customized to display the most common dosages and routes for that particular medication. If the medication is actually administered at a later time, the EMS personnel could simply press the "Set" button to log the current time, thereby relieving the EMS personnel of documenting in memory or on paper every time an action performed on the patient. Currently, however, all medication information must be manually entered. Near the bottom of the form is a Response to Treatment section in which EMS personnel can document the effects of each medication upon the patient. The Narrative page of the run record contains a free-form text area in which EMS personnel can type the narrative details of the incident scene, transport, treatment and treatment response of the patient as shown in Figure B-15 (Appendix B). The narrative becomes a legal document and, as such, cannot be automatically generated by the data captured by the Digital EMS system. It may be beneficial, however, to provide on this page an "incident timeline" that documents significant events, including dispatch and on-scene arrival time, time and names of administered medication, and hospital arrival time. Since the patient's vitals are constantly monitored and captured, the timeline could also document irregularities such as abnormal blood pressure readings, cardiac arrest, and respiration distress.

11.2.2 Paramedic-Mobile Station Video/Vitals Interface

Upon pressing the Video/Vitals button at the bottom of the screen, EMS personnel can immediately view video of the patient as well as vital patient data, as shown in Figure B-16 (Appendix B). Three on-board cameras have been integrated into the Digital EMS system, enhancing communication to the hospital by providing the physician on call a virtual "window" into the ambulance. As described later in this section, the physician has full rotation and zoom control over the three on-board cameras. The Video/Vitals screen enables the EMS personnel to view in real-time the exact camera angle, zoom, and resulting video that is being transmitted to the physician.

A Propaq vitals signs monitor (Protocol Systems, Inc.) has also been integrated into the Digital EMS system, enabling real-time patient vitals display, capture, and transmission. This enhances the EMS personnel's ability to not only review historical vitals data for post-incident review, but also provides timely communication of the patient's condition to the hospital physician. This Video/Vitals screen displays both the live video transmitted to the physician as well as the patient's vitals data.

11.2.3 On-line Protocols Interface

Upon pressing the "Protocols" button at the bottom of the screen, EMS personnel can immediately have access to a wealth of on-line medical procedure protocols, as shown in Figure B-17 (Appendix B). The protocol pages are implemented in standard hypertext markup language (HTML) format in which underlined words contain links to other pages. This enables the EMS personnel to simply press on the desired linked text to rapidly drill down to the desired medical protocol. Since this format is widely supported across software applications and the World Wide Web, additional protocols can be integrated quickly with little modification.

11.2.4 Map & Navigation Interface

On the Paramedic-Mobile Station, the moving vehicle is displayed at the center of the map. The EMS personnel station has the capability of calculating a route to destination based on whether it is a known location (a pull down menu of a list of hospitals, one of which can be selected), or an address. The address can be input in the form of a street address with or without zip code, a cross street location, X-Y or latitude-longitude coordinates. The Streets32 software is then used by the application to calculate and display a route. The Streets32 software generates a travel report that contains the estimated time to destination. The Paramedic-Mobile Station map interface displays the current location, destination, distance from destination and the estimated time of arrival at the destination. It also features a scrolling panel that EMS personnel can use to scroll the map in all directions. EMS personnel can switch between the scrolling mode (ambulance is not centered on the map) and the center mode (ambulance centered on the map). There is a pull-down menu that can be used to select the range (in miles) to be displayed on the map, and the map can be zoomed in or out according to the choice. A screenshot of the interface is shown in Figure B-18 (Appendix B).

11.2.5 Paramedic-Mobile Station Communications Interface

Upon pressing the "Comms" button at the bottom of the screen, EMS personnel will be able to view and modify parameters affecting data communication to the hospital, as shown in Figure B-19 (Appendix B). While this capability is currently under development, an explanation of its key features is provided here. The Intelligent Communications Manager (ICM) will automatically determine available bandwidth and reliability of all communication mediums at all times and coordinates their usage so as to maximize successful data transmission. From this screen, EMS personnel can view maximum and actual throughput values as well as change communication parameters and enable/disable a communication medium altogether. This screen is offered merely as a troubleshooting service to the EMS personnel, since communication will be optimized automatically via the ICM.

11.2.6 Run Record Report

The on-line run record, as discussed earlier in this section, is divided into a Run Record section and a Report screen. Upon pressing the "Report" button at the bottom of the screen, EMS personnel can view and modify time-insensitive billing and insurance information. It is important to note that all data entered in the report is automatically backed up with the run record, in several ways. Any time the user is switching to another run record page or to another system function, the system will automatically save the current page before switching. Also, the system utilizes a timed backup method in which the current page

is written to disk at specified intervals, such as every three minutes. The report data is not automatically transmitted to the hospital since it is not time-critical information.

The Report section is subdivided into five pages of forms. The first page, entitled Vitals Report, contains basic incident information, as shown in Figure B-20 (Appendix B) The bottom half of this screen is reserved for viewing historical Propaq vitals data which will be under development in the near future.The Billing page of the run record report contains responsible party and insurance information for the patient, as shown in Figure B-21 (Appendix B). The patient information row at the top of the form is automatically filled in, leaving the remainder of the form for EMS personnel to complete.

The Release page of the run record report contains the Release of Responsibility and Acknowledgement sections, as shown in Figure B-22 (Appendix B). With this form, the patient can acknowledge that medical care was or was not recommended and that he/she is denying care. This form and procedure will be submitted to legal counsel in the future to verify legality as a submission in a court of law.

The Transport/Crew page of the run record report contains patient transport method, authorizing physician, and destination of the patient, as shown in Figure B-23 (Appendix B). The Digital EMS system will be moving towards a secured logon EMS personnel and/or crews and, once in place, will enable the crew section to be automatically entered.

The Report page of the run record report will enable EMS personnel to preview or print the entire run record or only selected portions, as demonstrated in B-24 (Appendix B). While this capability is currently under development, an explanation of its key features is presented here. This form contains toggle checkboxes that enable EMS personnel to quickly select/deselect any of the pages from both the Run Record and Report sections. The resulting printout will contain all textual data and images from the selected sections in a modular and concise format, emulating many of the run record forms in use today. Once this capability is implemented, it is expected that the EMS personnel can send the printout either to a hospital printer or one incorporated into the on-board Digital EMS system.

11.2.7 Computer-Based Training Interface

Upon pressing the "Training" button at the bottom of the screen, EMS personnel will be able to receive computer-based training (CBT) on board the ambulance, as demonstrated in Figure B-25 (Appendix B). While this capability is currently under development, a variety of interactive medical training courses can be integrated into the Digital EMS system that utilize 2D and 3D illustrations, video, voice, testing, and feedback to provide a professional curriculum and meet national standards of the Department of Transportation. Integrating interactive, multimedia courses enables EMS personnel to analyze case studies of real emergency calls, refresh and test their knowledge of EMS protocols, or consult an on-line medical dictionary at any time the ambulance is powered up or en route.

12 Communications Infrastructure

The Digital EMS project data requirement includes real-time video, audio, physiological data (vital signs) display, and patient medical information access (data transfer). Since the system can roam anywhere relative to the hospital (urban or rural), a radio frequency (RF) data infrastructure that has high coverage and high data throughput must be employed. The capacity requirements for the communication system will be lower bounded by the throughput required to support the vital signs data. In addition, for compatibility, the communication protocol between the ambulance and the hospital's sub-network is

TCP/IP and UDP/IP such that standard operating system drivers and interfaces might be used. As an additional requirement, the RF infrastructure for the project must support wireless Internet protocol (IP). Currently, DIGITAL EMS supports the following IP based radios: cellular digital packet data (CDPD), Motorola's DataTAC2.0 and Breezecom's BreezeNET DS.11 wireless LAN for the communication paths.

Responsibility for satellite research is shared between The Texas A&M University System Digital EMS and the University of Texas Health Science Center at Houston. Areas under investigation include commercial VSAT and Ku systems, experimental and future commercial Ka LEO and geosynchronous, as well as systems originally designed for entertainment and Internet services.

Since Digital EMS supports multiple communications systems, an Intelligent Communications Manager (ICM) is under current development to manage each communication path. The ICM will oversee and control packet transmission as a function of available bandwidth, data priority, channel usage cost, and real time channel degradation. The following communication infrastructures were considered.

12.1 Communication Technologies

Many different communication technologies exist that support wireless IP; some are categorized below:

- Low Capacity Communication Technologies
  - Cellular Technologies.
  - CDPD.
  - Metricom Ricochet.
  - ARDIS.
  - PCS.
  - CDMA.
  - GSM.
  - GPRS.
  - TDMA.
  - Motorola DataTAC 2.0 (Privately owned by customer).
  - QualComm (Owned by service provider).
  - Ericsson (Owned by service provider).
- High Capacity Communication Technologies
  - Point to Point.
  - Wireless LAN.
  - Wi-Linx wireless LAN.
  - Lucent.
  - GRE America.
  - Mobile Radio.
  - BreezeCom.

The following tables summarizes each of the aforementioned technologies capabilities:

Low Capacity Wireless Technologies

Technology	Channel Data Rate (Kbps)	Throughput (Kbps)	Encryption	MobileEQ	Coding	College Station	Houston
CDPD	19.2	10-12	Yes	Yes	Yes	Yes	Yes
Metricom	100	28.8	No	No	No	No	No
ARDIS	19.2	9.6	No	No	Yes	No	No
CDMA	14.4	9.6	Yes	Yes	Yes	No	No
GSM	9.6	?	Yes	Yes	Yes	No	No
TDMA	9.6	?	No	Yes	Yes	No	No
Analog Cellular	9.6	4.8	No	Yes	Yes	Yes	Yes
DataTAC 2.0	19.2	9.6	Yes	Yes	Yes	Yes	Yes
QualComm	19.2	14.4	Yes	Yes	Yes	No	No

References:

AT&T white papers WHT002 Rev 1.2 (CDPD, Digital Cellular and PCS Networks)
AT&T white papers WHT004 Rev 1.1 (Wireless Data Networks Comparison)

High Capacity Wireless Technologies

Technology	Channel Data Rate (Mbps)	Throughput(Mbps)	Encryption	Mobile EQ	Distance	Speed (mph)	Coding
BreezeCom	11	10	No	Yes	2miles	60	Yes
Wave Point Lucent Tech	10	?	Yes	No	25 Miles	0	No
M1011 Clarion	10	?	No	No	25 Miles	0	No

References:

http://www.breezecom.com
http://www.lucent.com/enterprise/brochures/wir0680.pdf
http://www.karlnet.com/products/kits/radios.html

12.2 Wireless Communications Technology Feasibility

The decision metric for choosing a particular system is a function of many parameters such as throughput, availability, reliability, compatibility, coverage area, quality of service, privacy and cost. For example,

many high-end systems such as Motorola's DataTAC 2.0 require the end user to purchase all required hardware including base station, antenna and tower, network controller, message switch, and mobile radios as well as software and FCC licensing for spectrum usage. In many cases (police department and fire department), owning the communication infrastructure is necessary to establish one and to guarantee a specified quality of service, effective data throughput and privacy. Of course, the cost of such a system will be considerably high ($400,000) and the coverage will be constrained to that of private towers. Another approach is to look at what service is available through a provider such as AT&T, BellSouth etc. To illustrate, AT&T provides a mobile wireless IP service referred to as cellular digital packet data (CDPD) that can provide encrypted Internet connectivity via the advanced mobile phone system (AMPS) for a flat charge of $50 per month roam free. Accordingly, this system is an ISP and the quality of service and effective data throughput is a function of the number of registered users.

12.3 Satellite Communication Technology Feasibility

Satellite research for Digital EMS is shared between The Texas A&M University System and the University of Texas Health Science Center at Houston. UT-Houston has established a relationship with the Naval Research Laboratory in Washington, DC. In conjunction with TEES, UTHHSC is directing the NRL in the development of an ambulance mounted dish antenna capable of tracking a low earth orbit (LEO) satellite while the ambulance travels at 70 mph on the highway. The equipment is on order. The initial prototype is Ku band. A later phase will use Ka band. At that time, we expect to test with prototype commercial satellites and perhaps the NASA Tracking and Data Relay Satellite System (TDRSS) satellite service.

NASA's Glenn Space Center will support an upcoming demonstration of the Digital EMS ambulance over the ACTs Ka band satellite. The ambulance will be in a remote location connected to a portable USAT (Ultra Small Aperture) antenna. The other downlink will be at the Naval Research Laboratory in Washington, D.C. From there, the link will travel over the DREN (Defense Research Engineering Network) and the Internet2 backbone to the Houston Digital EMS facility in the Texas Medical Center. A medical triage demonstration including vital signs, video conferencing, and remote equipment control will be presented.

Other areas of satellite investigation include the potential of using commercial entertainment and Internet satellite mounts and antennas designed for recreational vehicles and boats. These have a large potential consumer market and represent an affordable solution if enough effective bandwidth can be obtained. Examples are products from Dat-com, Winegard, and Direct PC. Equipment is on order to evaluate different possible solutions. The most promising antenna is an experimental prototype from Raytheon Industries weighing less than 20 pounds and approximately 6"H x 12"W x 30"L in size. This is an electronically and mechanically steered device. Raytheon is targeting 2 mbps for less than $2000.

12.4 Digital EMS Currently Supported Technologies

The Digital EMS system currently supports CDPD, Motorola's DataTAC 2.0 and Breezecom's BreezeNET DS.11 wireless LAN. CDPD was chosen due to its wide coverage area, data throughput and low cost. The average throughput is 12.5 kbps and since video compression is currently employed and the ICM supports multiple devices, using four CDPD modems can support the current video capacity requirement. The Motorola radios are supported to be compatible with local police and fire department communication infrastructures. Satellite will be supported when technology that suites the needs of

Digital EMS becomes available. The BreezeNET DS.11 is supported for its short-range ability to give multiple mbps to a noisy vehicle.

12.4.1 CDPD

CDPD is a broadly available wireless packet data network system that overlays the existing ubiquitous Advanced Mobile Phone Service (AMPS). CDPD modems support data rates up to 19.2 kbps, are serially connected to a computer via RS232 and communicate via PPP/SLIP. The setup time is approximately one second to establish a connection. User data is received by a cell tower and is then via wire line routed to the CDPD AirData Network. User packets are then routed via the Internet to the appropriate IP address as shown in Figure 12.4.1-1.

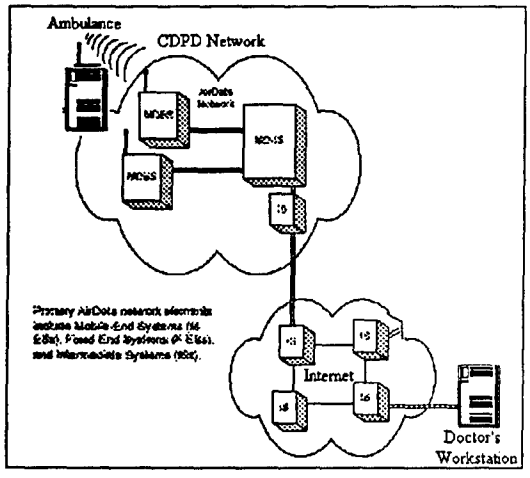


Figure 12.4.1-1. CDPD RF Infrastructure.

Figures 12.4.1-2 and 12.4.1-3 show the existing coverage for CDPD services in the United States and for the local Houston metropolitan area.

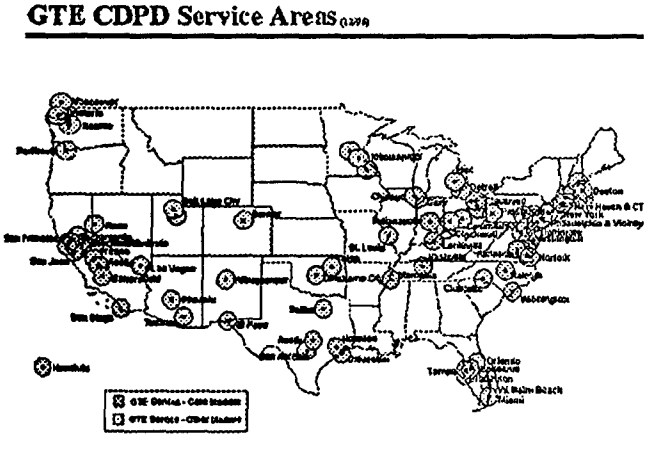


Figure 12.4.1-2. CDPD Coverage in the United States.

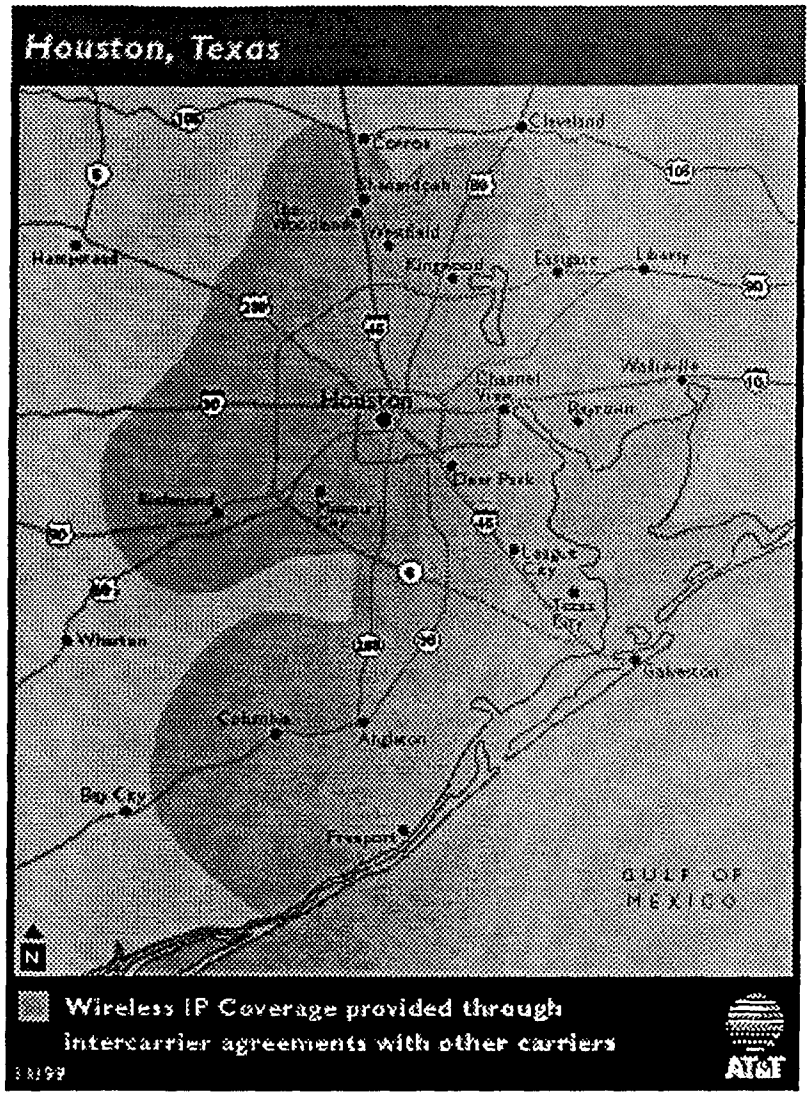


Figure 12.4.1-3. CDPD Coverage of Houston.

12.4.2 Motorola DataTAC 2.0

DataTAC2.0 is a proprietary wireless packet data network that is commonly used by public safety departments (police, fire, EMS). Motorola's modems support data rates up to 9600 bps, are serially connected to a computer via RS232, and communicate via PPP/SLIP. User data is received by a DataTAC data base station and is then routed to the Remote Network Controller (RNC) via optical fiber or wireline. User packets are then routed via the Internet to the appropriate IP address as shown in Figure 12.4.2-1.

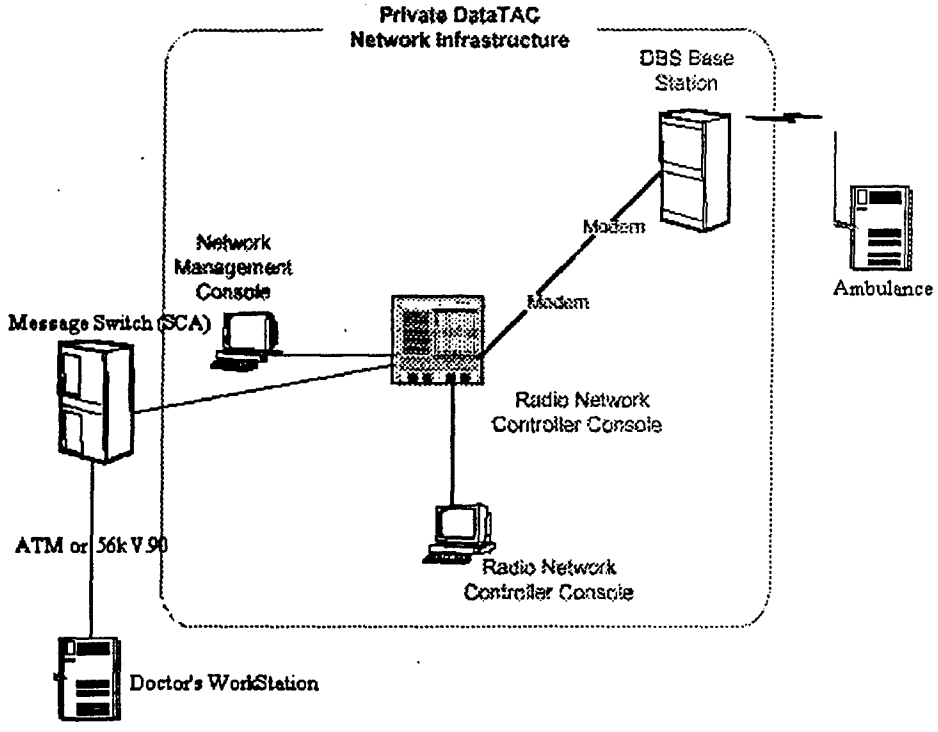


Figure 12.4.2-1. Motorola DataTAC 2.0 RF Infrastructure.

12.4.3 BreezeNET

The BreezeNET DS.11 high speed, wireless network is an IEEE 802.11-compliant product line designed to provide indoor client/server network configurations as well as long range point-to-multipoint links. The products use direct sequence, spread spectrum radio technology operating at a frequency of 2.4–2.4835 GHz, a part of the FCC's unlicensed Industrial, Science, Medical (ISM) band. Data is transmitted at a rate of 11 Mbps, providing network users with full 10BASE-T Ethernet speeds. The DS.11 product line is comprised of two distinct product line sets: one for indoor client/server and, another primarily for outdoor, longer network-to-network and client-to-network connections as shown in Figure 12.4.3-1.

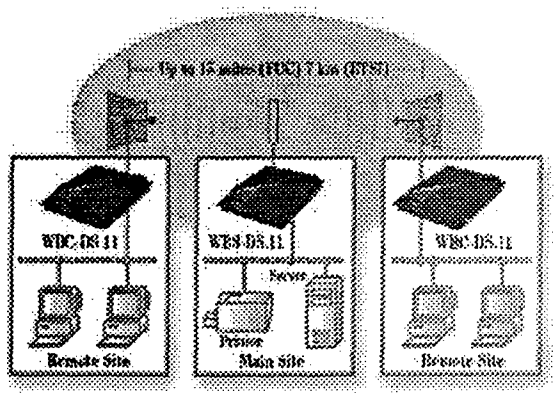


Figure 12.4.3-1. BreezeCom RF infrastructure.

12.5 Middleware and Interfaces

The middleware is a set of APIs that are used by the server and client applications to communicate with each other and the Intelligent Communications Manager (ICM). The middleware is used as an interface between the applications and the underlying network layers as shown in Figure 12.5-1. This abstracts TCP/UDP and IP from the applications. The implementation details of a connection can be hidden from the applications that need to have no knowledge of these details. The middleware provides more control over the transport and network layers. The middleware uses datagram (UDP) sockets to establish connections. He choice of UDP allows better visibility into network conditions and control over bandwidth allocation. TCP is a connection-oriented protocol and, hence, more reliable than UDP, which is connectionless. This can be overcome by adding the necessary level of reliability to UDP in the middleware and have as much control as needed over the transmission protocol. Timeouts on the receiving end, acknowledgements, and retransmissions will be used to add the needed level of reliability to UDP.

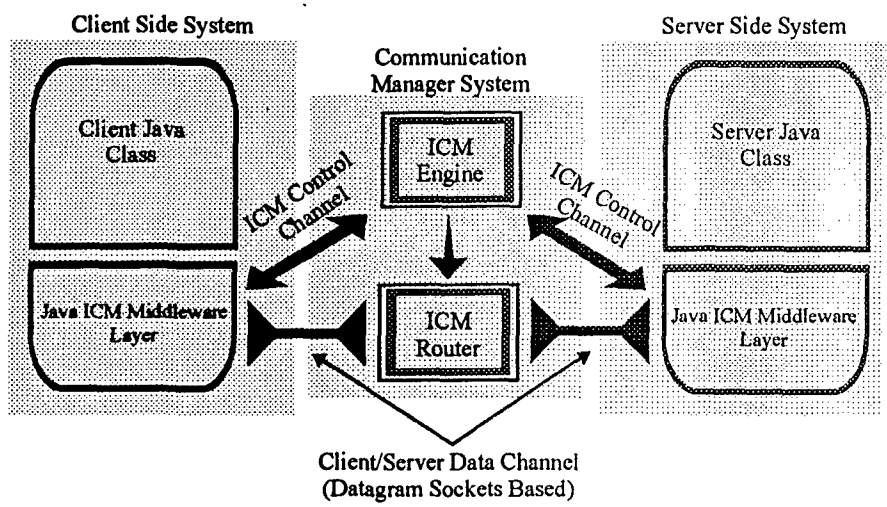


Figure 12.5-1. ICM module components.

The ambulance systems (video and control servers and Paramedic-Mobile Station) and the hospital system (Physician Stations) may have one or more applications. The applications use the APIs to initiate a connection, transfer messages and close connections. The wireless connections and devices are managed by the ICM. The middleware is written in Java, as are the servers and clients that execute in the closed Digital EMS system. Middleware objects are implemented as a single middleware class that implements the communication APIs required to access data transfer and ICM functionality. Each application (client and server) initiates a middleware object during the initialization phase before any communications can be done. When each application creates its middleware object, it specifies a unique ID and type (server or a client). Communications is then established to the ICM module through the use of a set of well known ports which are mapped to all clients and servers transparently. The middleware has the list of port numbers assigned to the applications, which it uses to create sockets. The APIs in the middleware are explained in the following paragraphs.

12.5.1 Middleware API

The following are the APIs provided by the middleware for the applications to communicate with the ICM:

- Register

This method allows server applications to register with the ICM so that clients that need to connect to a particular server can do so. All server applications use this call to register themselves with the ICM. The middleware creates a server socket at the port assigned to the particular server ID. It returns an integer denoting success or failure to the server. The flow of the server registration call is shown in Figure 12.5.1-1.

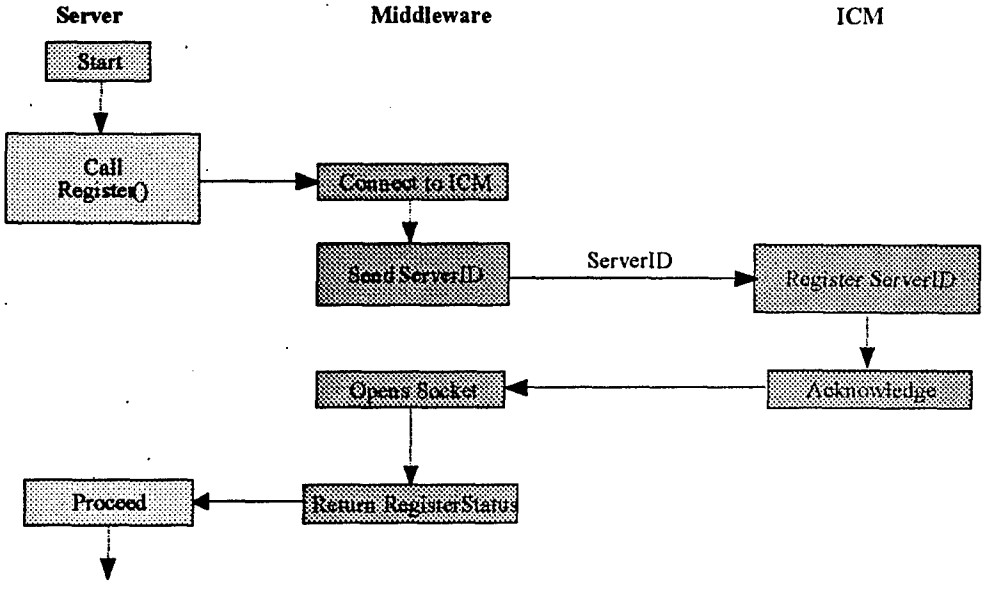


Figure 12.5.1-1. Flow of Register method.

- ConnectRequest (Server, data type, priority, reliability, required bandwidth, connectionID)

This method allows clients to request a connection to a particular server. The client requests a connection to the desired server, specifying the data type (video frame, vital sign data, vital signs waveform, run record data etc.), priority, reliability, required bandwidth, and a connection ID unique to this connection. The middleware connects to the ICM, which determines if a connection is possible based on whether the server has registered with it and the priority and bandwidth requirements of the client. When this is determined, the ICM informs the middleware of the decision. The middleware then creates a socket for the client to listen on and receive incoming data and returns connection status (success or failure) to the client based on the ICM's decision. At this point, the client is ready to send a request to the server for the data it needs. Figure 12.5.1-2 illustrates the flow of the ConnectRequest call.

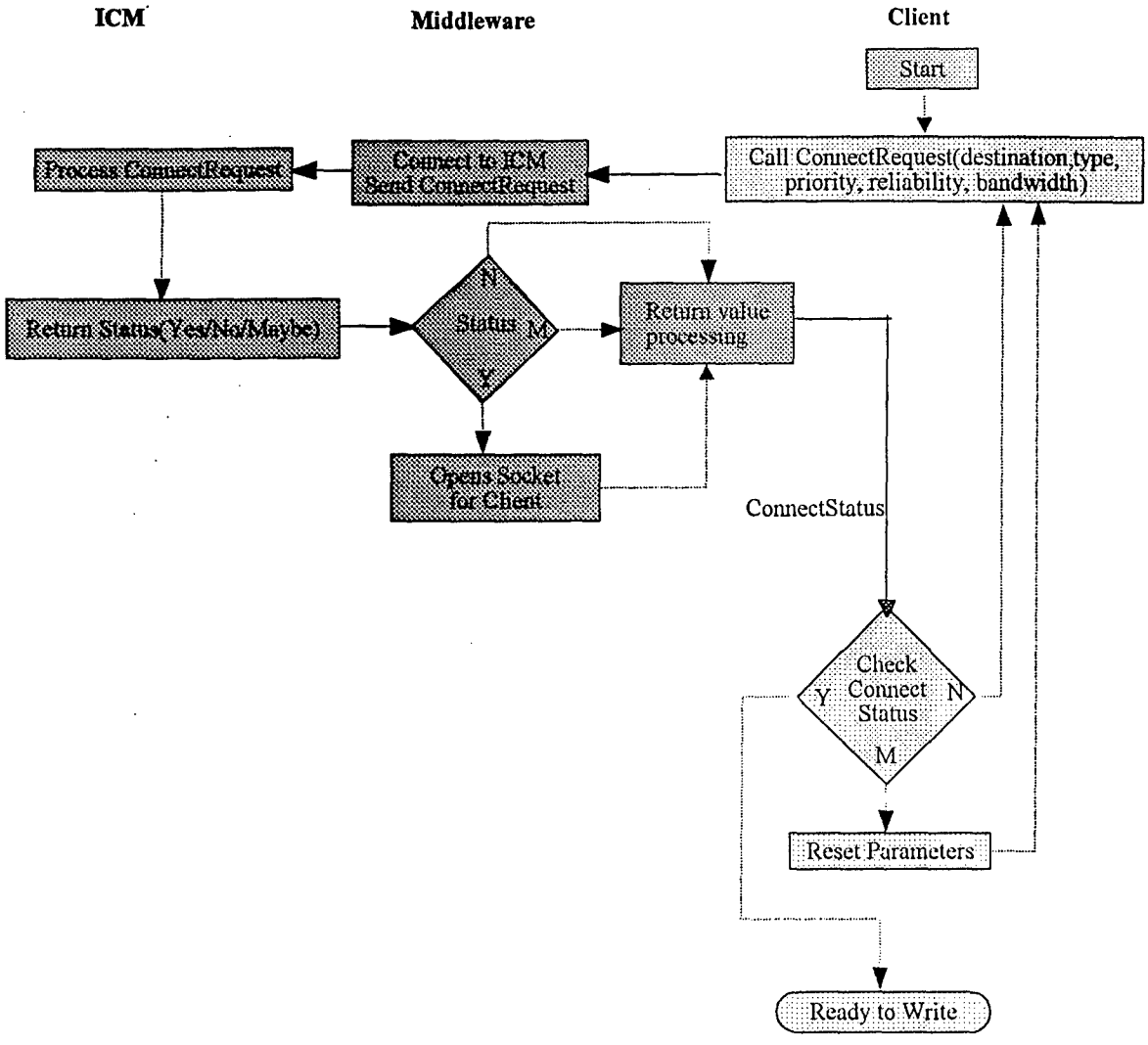


Figure 12.5.1-2. Flow of ConnectRequest method.

- Accept

This function allows servers and clients to listen on a socket and wait to receive from a client or server. This is a blocking call until something is received on the socket. This method returns when something is received. Once the servers have registered, they call this method and wait for a request to be received. The clients do the same when they have sent a request to the server and need to wait for data.

- Read

This feature lets servers and clients read the incoming information on the socket, reads the datagram received during the Accept call, and returns an array of bytes read. If multiple packets are received, they may need to be reordered before being returned to the application. When a server reads a request from a client, it processes the request and sends the data to the client (via the ICM). The middleware handles the timeouts and determines if a retransmission is required so that the communication is reliable. Figure 12.4.1-3 shows the data flow between the server and client.

- Write (command/data)

The Write method lets servers and clients write data. The middleware sends a datagram over the socket to the ICM, which is listening to all the servers and clients and returns a success or failure. The middleware packets the data if necessary, and sends multiple times depending on the amount of data that needs to be sent. The ICM receives the data and sends the data to the server or client, which constitutes the other side of the connection. As described earlier, the ICM serves to transfer data between the servers and clients. For each connection, the ICM establishes two listening sockets, one on the server side and one on the client side. It transfers service requests from the client to the server, and data from the server to the client. Since UDP does not guarantee sequential delivery, the middleware on the receiving side may need to reorder the packets and deliver them to the application. See Figure 12.5.1-3 for the data flow between server and client.

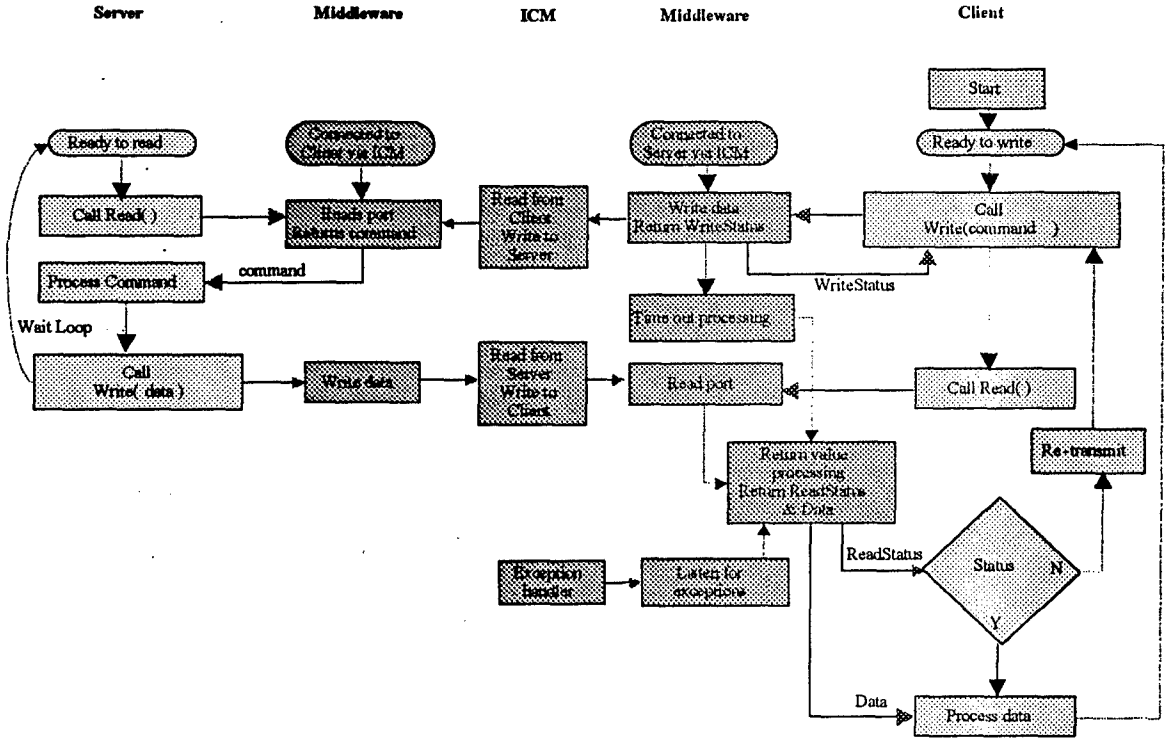


Figure 12.5.1-3. Flow of Read and Write methods.

- QueryChannel

This feature allows an application to query the parameters like maximum and effective throughput, available bandwidth etc., of an available channel. Based on these values the application may make decisions on channel requirements.

- Close

This function allows an application to close a connection.

The middleware also provides a Reliability class that can be used by the applications to request the level of reliability the connection needs. It consists of two variables, reliability level and buffer time. The level is a numeric value that sets the level of reliability needed by the application and buffer time indicates how long the data needs to be buffered before it can be discarded. Some types of data streams need not be

buffered, and some of them need to be indefinitely buffered. By setting these variables, the applications let the ICM know how the data streams need to be managed.

The priority of data is another parameter the applications set, and this determines how the ICM prioritizes data from the application. The ICM will be able to reset or reorder the priorities depending on the channels available and the physician's input.

12.6 Intelligent Communications Manager (ICM)

The Intelligent Communications Manager (ICM) is a software module that manages the wireless communication links and dynamically allocates the available bandwidth depending on the type and priority of the data that must be transmitted. The wireless devices that can be deployed support varying data rates. For example, the CDPD modems have a maximum throughput of 19.2 kbps, while the Motorola data radios have a maximum data rate of 9.6 kbps. The ICM supports varying data rates on multiple channels and over heterogeneous devices. The ICM functionality allows the available channels and bandwidth to be treated as one entity, relieving the applications of the management burden. The bandwidth available to the applications keeps changing because the wireless links may become available and unavailable. The ICM makes this largely transparent to the applications. The applications still have some control over the channels in that they can query the ICM periodically to determine what bandwidth is available and alter their transmission parameters accordingly. The applications may also restrict the channel on which they want their data to be transmitted. However, the intelligence primarily resides in the ICM, which manages the links and traffic at all times, performing load balancing and failover without explicit application awareness. Features include the following:

- It has its own protocols for handling links failing and becoming available again.
- It keeps track of all the devices that are available at any instant, and the effective throughput available.
- It matches the bandwidth and channel requirements of the applications with the bandwidth available.
- If it is not possible to provide the bandwidth requested by the applications, it informs the applications accordingly, which then modify their requirements.
- It has the capability to buffer data depending on the reliability requested by the applications.

Applications can request various levels of reliability and different buffering times. If maximum reliability is needed, the ICM buffers all the data until it can be sent to the physician. If partial reliability is needed, data is buffered for the buffer time requested. If data need not be reliable, it is not buffered. The ICM handles queuing based on the level of priority requested. A knowledge base will be implemented that will support graceful degradation of the system. Factors such as channel cost, data priority, available bandwidth and delivery time will be used in decision-making. Further, the ICM will have functionality to override the knowledge base with the physician's requirements. For instance, the satellite channel will be used regardless of channel cost if the physician requires high-resolution video.

The ICM consists of program(s) executing on the onboard Linux system and middleware packages on each workstation. The middleware further insulates applications from the communications details. Linux was chosen due to its system due to the stability and superior networking and multitasking capabilities of this operating environment. The systems on the ambulance will be on a local subnet and communication will be through 100 Base-T Ethernet. Communication between ambulance and hospital network will be via wireless devices - Motorola DataTAC, CDPD and satellite. The communications system computer has

an integrated RocketPort, which provides multiple serial ports (RS232) for interfacing with Motorola data radios and CDPD modems. Interface for the satellite link will be a Network Interface Card (NIC).

Figure 12.6-1 shows the setup of the ICM on the ambulance. S1, S2, ... ,SN are rack mounted computer systems on which the applications reside. The machine labeled Communications Processor acts as the ICM. This machine also serves as the gateway (router) for the other machines on the ambulance since they are on a local subnet within the ambulance. These machines are connected to a 100 Base-T hub. The wireless modems are attached to serial ports on the communications processor. Four modems are shown in the figure, but there could be more depending on the data requirements and radio technologies. The satellite link is connected to a NIC on the communications processor. The antennas of the devices are mounted on the roof of the ambulance.

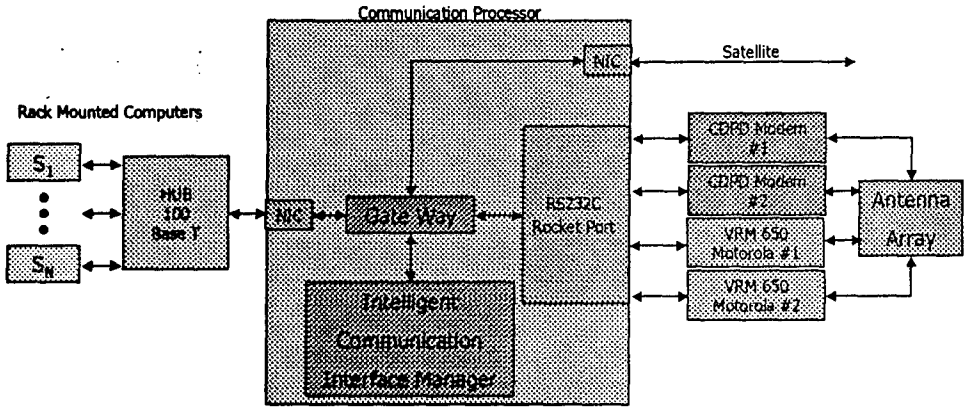


Figure 12.6-1. Communications system setup on ambulance.

The software development of the ICM is being done in stages. The first version simplifies certain aspects of the ICM in that it merely serves to transfer data between the servers and the clients through predetermined ports. The ICM reads a file, which has details of all the connections in the system and the port numbers associated with them. For each connection in the file, the ICM forks a process, which creates two sockets, one on the server side and the other on the client side and listens to the applications on these two sockets. When something is received on one of these listeners, it sends it to the application on the other side for processing. Commands or requests from the client are sent to the server, and data received from the server is sent to the client. In this version of the ICM, data streams are not split over multiple channels. If a link fails during transmission, the stream is diverted to another available channel. Multiple streams can be sent over one channel. The serial device interfaces have IP addresses, and each listener process has a local IP and port and a remote IP and port. The socket is bound to the physical device, and data on that socket will be transmitted on the device associated with this particular port. The ICM interfaces with the applications through the middleware.

Each new version of the software will add additional capabilities until it can provide all the requirements of the ICM. Data streams will be multiplexed over multiple channels. The packet number will be embedded in the headers to be de-multiplexed and reordered on the receiving side. Allocation of local resources (port numbers etc.) will be dynamic, and the applications will query the ICM to obtain connection information for requesting communications. The ICM will prioritize data based on the scenario or the physician's requirements. The cost function calculation will be implemented, which will be used by the ICM to make decisions on channels to use.

12.6.1 ICM Test Procedures

The test plan for the ICM is detailed below:

- Phase 1

The ICM sends data between servers and clients on two different subnets using NICs (Figure 12.6.1-1). Applications interface with the ICM via the middleware APIs. Applications communicating using Java RMI are being re-written to communicate using sockets, and being tested on different subnets.

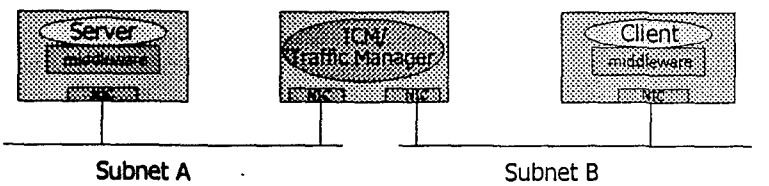


Figure 12.6.1-1. ICM Test Plan Phase 1.

- Phase 2

In this phase of testing, a computer with multiple interfaces (serial ports and NIC card) will be used to simulate wireless links (Figure 12.6.1-2). The ICM will send data from server to client using the NIC (high bandwidth) and serial cables (low bandwidth). Ethernet will be used to simulate the high bandwidth channel (BreezeCom wireless LAN and Satellite). Serial connections will be used to simulate the low bandwidth channels (CDPD modems and Motorola Data radios).

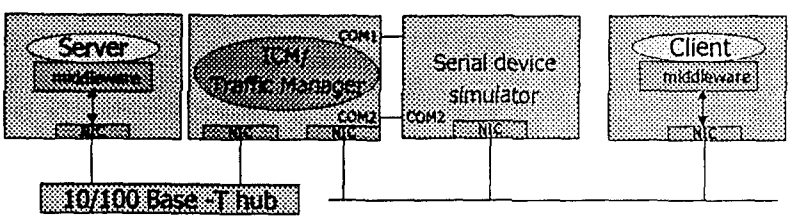


Figure 12.6.1-2. ICM Test Plan Phase 2.

- Phase 3

In the last phase of testing, wireless equipment will replace computer simulating the links. The ICM sends data from server to client using the BreezCom wireless modem and CDPD modems. This is shown in Figure 12.6.1-3. The satellite link to the system will be added when it becomes available.

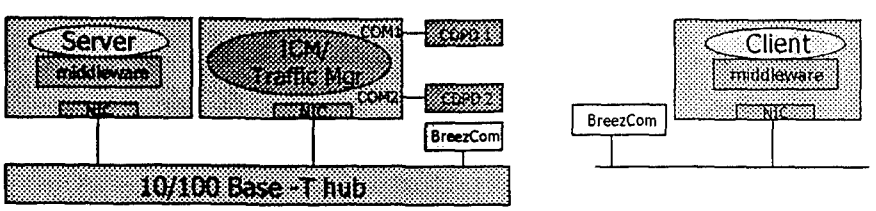


Figure 12.6.1-3. ICM Test Plan Phase 3.

13 Emergency Vehicle Modifications

The Digital EMS vehicle was modified to accommodate the installation of the following subsystems.

13.1 Computer Systems Mount

The computers installed are heavy-duty rack mountable systems. The systems are mounted on a standard industrial rack and installed in the radio compartment of the ambulance. The rack was mounted on tuned spring mounts as shown in Figures 13.1-1 and 13.1-2.

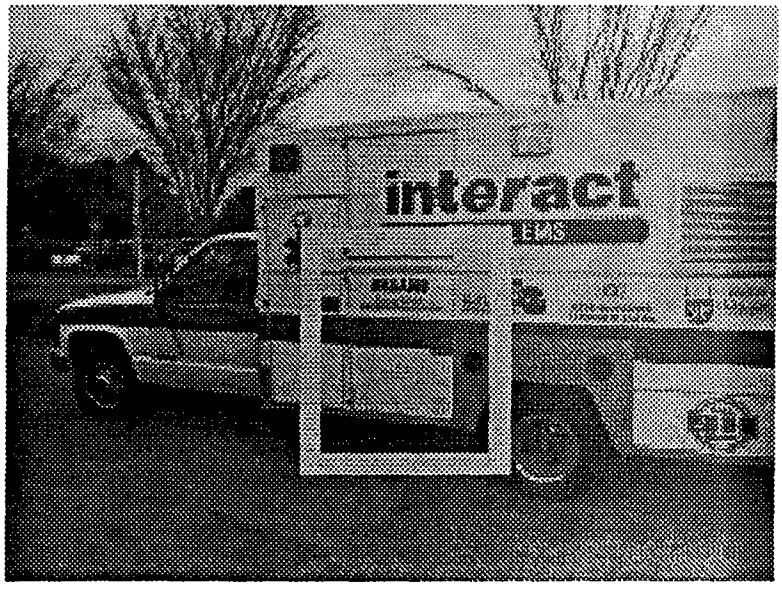


Figure 13.1-1. Digital EMS Ambulance Radio Compartment.

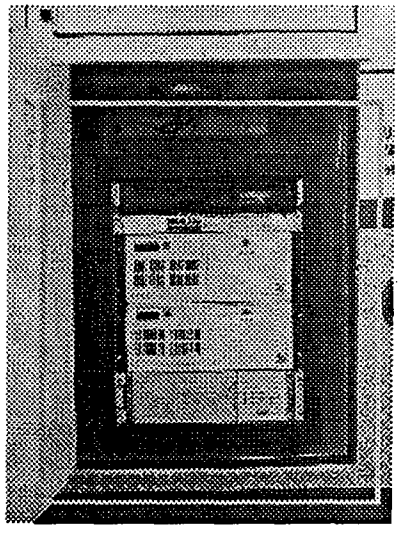


Figure 13.1-2. Rack mount system (mounted on vehicle).

13.2 Medic-Cam

The Medic-Cam system, as shown in Figure 13.2-1, is a small, lightweight, mobile video and audio coder/decoder used by medical staff to provide expert medical assistance remotely relative to a mobile base station (ambulance).

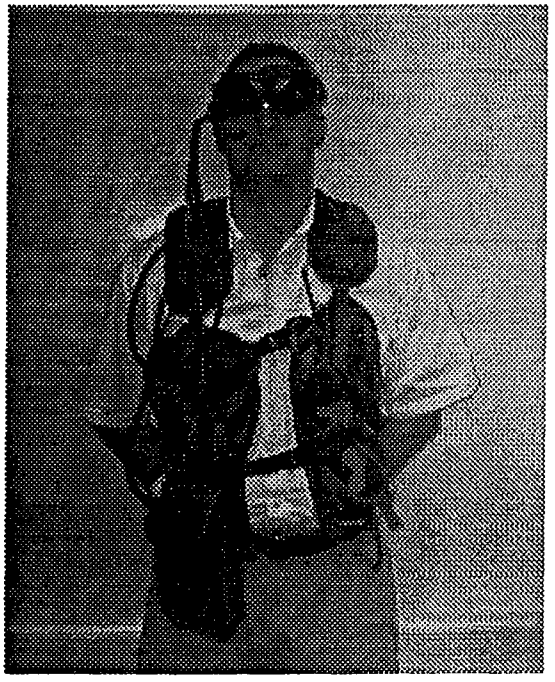


Figure 13.2-1. Medic-Cam system.

13.3 Video Cameras

The Sony EVI-D30 cameras are used to view the patent from two independent perspectives and are mounted on quick-release customized brackets mounted on the ceiling of the ambulance. The cameras were mounted to the middle right and bottom left of the ambulance patient compartment as shown in Figure 13.3-1. Cameras are mounted on the roof of the ambulance cabin as shown in Figure 13.3-2. Further analysis of the camera position will be carried out.

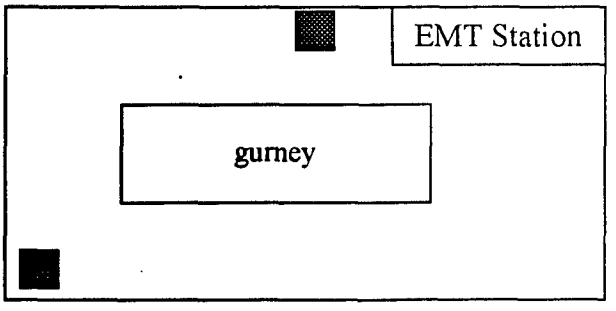


Figure 13.3-1. Camera configuration.

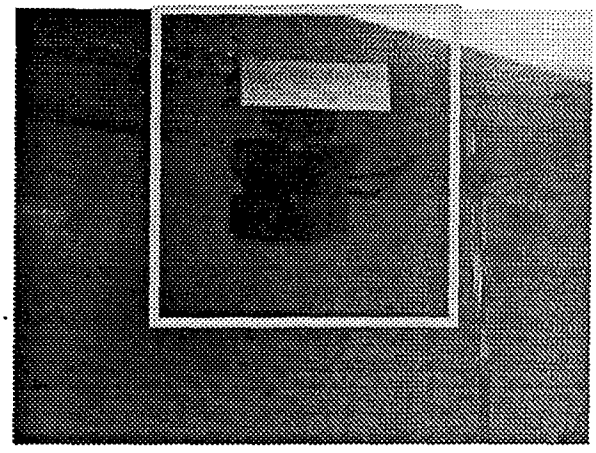


Figure 13.3-2. Sony EVI-D30 Camera (mounted on roof of cabin in vehicle).

13.4 IO Device Mounts

Several mechanisms have been developed to attach the various components to the ambulance. The following sections describe these techniques.

13.4.1 Keyboard and Monitor

The monitor is a 15" flat touch-screen by Microtouch, Inc. and is mounted on spring shock mounts similar to the rack. The keyboard is hermetically sealed and has a touch pad to emulate a mouse. Part of the radio compartment was used to build an enclosure for the keyboard and Plexiglas™ was placed above as to provide additional EMS working surface (see Figure 13.4.1-1).

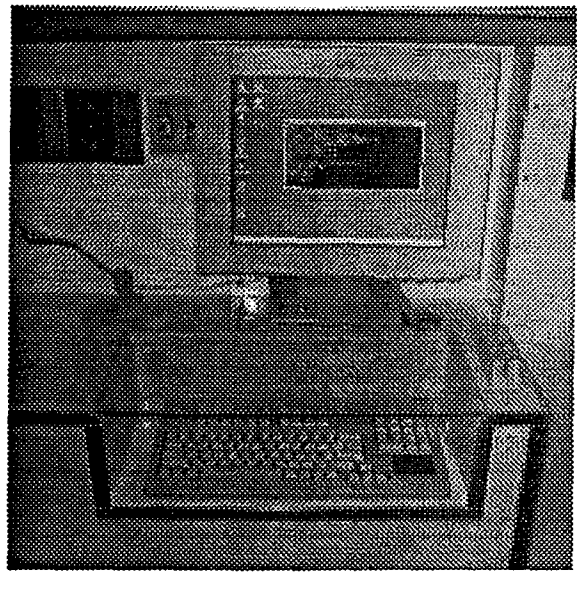


Figure 13.4.1-1. Monitor, keyboard, and microphone in vehicle.

13.4.2 Propaq Vital Signs Mount and Speaker

The Propaq Encore by Protocol Systems, Inc. was integrated into the Digital EMS project to provide vital signs data (heart rate, blood pressure) to the Physician Station. This device was mounted to the left of the EMT workstation on a aluminum rugged swivel mount. Data and power connections were routed via the radio compartment bulkhead to the rack-mount computer systems.

The speaker was mounted below and to the left of the EMT workstation. This device is driven by a SoundBlaster card in rack mount computer two (Ambulance 2) via a 40 watt audio amplifier.

13.5 Ambulance Antennas

Three independent communication paths are currently supported: DS-11 BreezeNet, CDPD and Medic-Cam. The DS-11 supports two antennas for diversity and CDPD supports one. The Medic-Cam also supports one. Each antenna required holes to be drilled on the roof of the box and additional weather proofing material as shown in Figure 13.5-1.

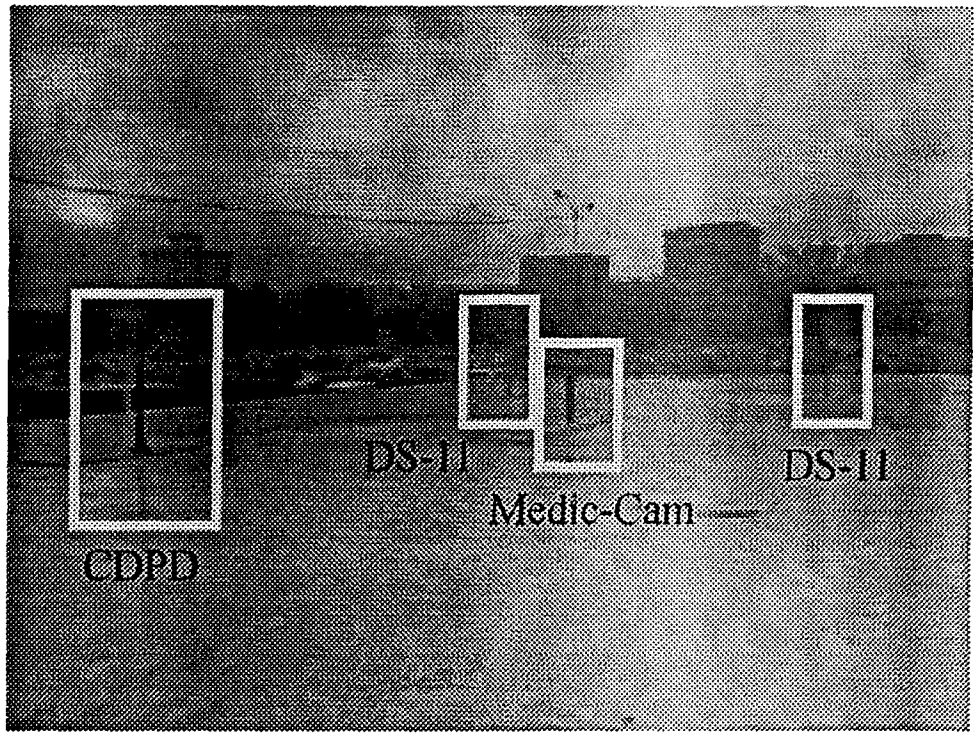


Figure 13.5-1. Antenna layout on vehicle.

14 Army Field Ambulance Feasibility Study

Current work on the Army's field ambulance is focused on feasibility studies to investigate requirements for transferring Digital EMS technologies to a U.S. Army field ambulance. Current Army field ambulances consists of an HMMVV vehicle outfitted with a hard shell on the back for patient transport as shown in Figures 14-1 and 14-2.

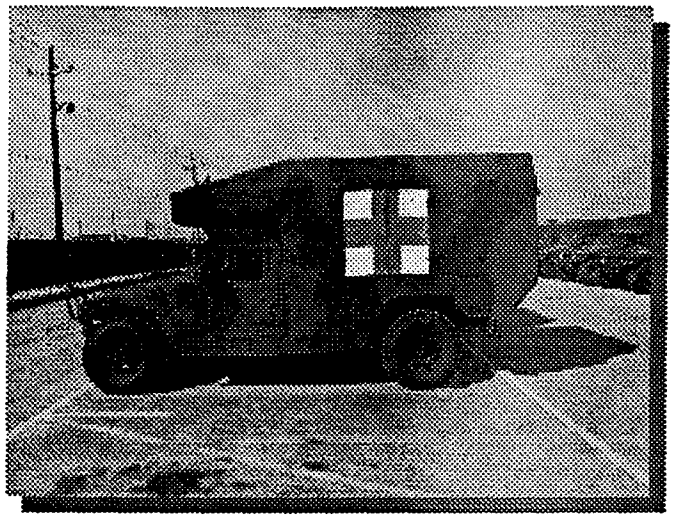


Figure 14-1. Outside field ambulance configuration.

Figure 14-2. Interior configuration.

Issues must be resolved for computer system mounts, antenna mounts and communication, interface design and development, IO device development, and mechanical/electrical vehicle modifications. Substantial modifications must be implemented to the back mechanical infrastructure to accommodate the various systems required for full Digital EMS functionality.

15 Conclusion

his report provides an overview of the design and implementation work accomplished for the Digital EMS project at the Texas A&M University System under subcontract DAMD 17-98-2-8002. These efforts consisted of the overall system design and basic proof of concept implementation of some of the system functions required to implement basic interface and system wide architecture tasks for the Digital EMS vehicle and base station.

Future directions for this project include multiple additions and enhancements to the current system design of both the hospital and on board vehicle subsystems. These include the addition of support for other medical diagnosis equipment, additional communication infrastructure support, and additional system functionality.

One aspect of the current work will focus on adding system functionality for better remote support of emergency personnel while outside the vehicle through the use of wearable computer systems. This creates localized independent processing support that can be utilized by loadable Digital EMS software. Local software modules will allow outside paramedics to initiate and begin the patient treatment process without the need to have the patient on board the vehicle. Additionally, this will also allow for a better and more efficient triage process when dealing with multiple casualties outside the immediate area of the remote vehicle.

Future work will also concentrate on improving the communication system to allow for applications to make full use of the Intelligent Communications Manager. Work will focus on improving the scheduling and processing of communication links between the remote vehicle and base station through the use of better heuristics and algorithms for link scheduling and multiplexing. The ICM will be able to adjust communication links based not only on the data requirements, but also on a communications cost function which attempts to maximize required bandwidth based on the available communication devices, type of situation, and physician requirements.

Application development improvements will be geared towards better physician and EMS personnel interfaces along with better support for additional medical and non-medical devices. Functionality will be added for establishing database connectivity from the remote vehicle to local hospital databases. In this manner, run record information tags will be extracted for inclusion on the paramedic station on board the vehicle. Application interfaces will be enhanced to accommodate better support for training, navigation, and run record functionality in addition to automated run record creation through medical device communications.

Overall system design is moving to implement a full turnkey system that can be used operationally with minimal administrative support for setup and execution. In this manner, emergency personnel will be able to operate the system on field test measurements with proper Digital EMS user training.

Appendix A: Computational Flow Diagrams

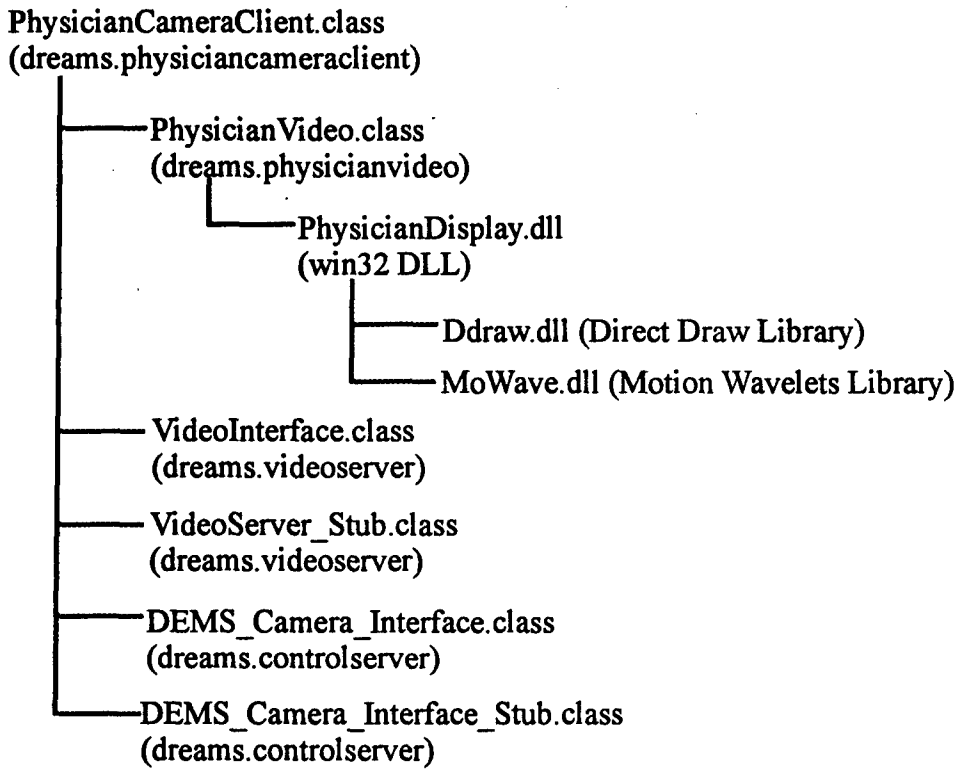


Figure A-1. Dependency diagram for main Physician Camera Client class.

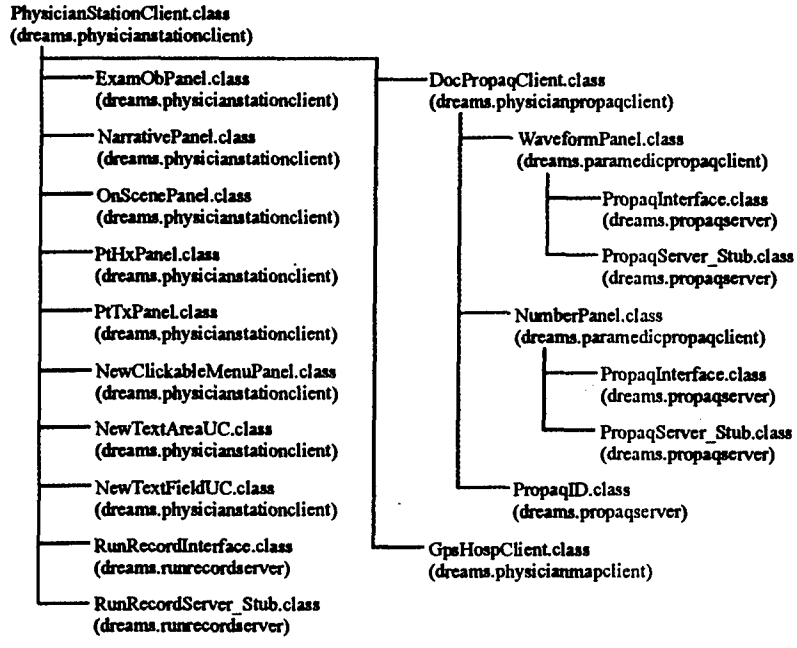


Figure A-2. Dependency diagram for Physician Station Client Java class.

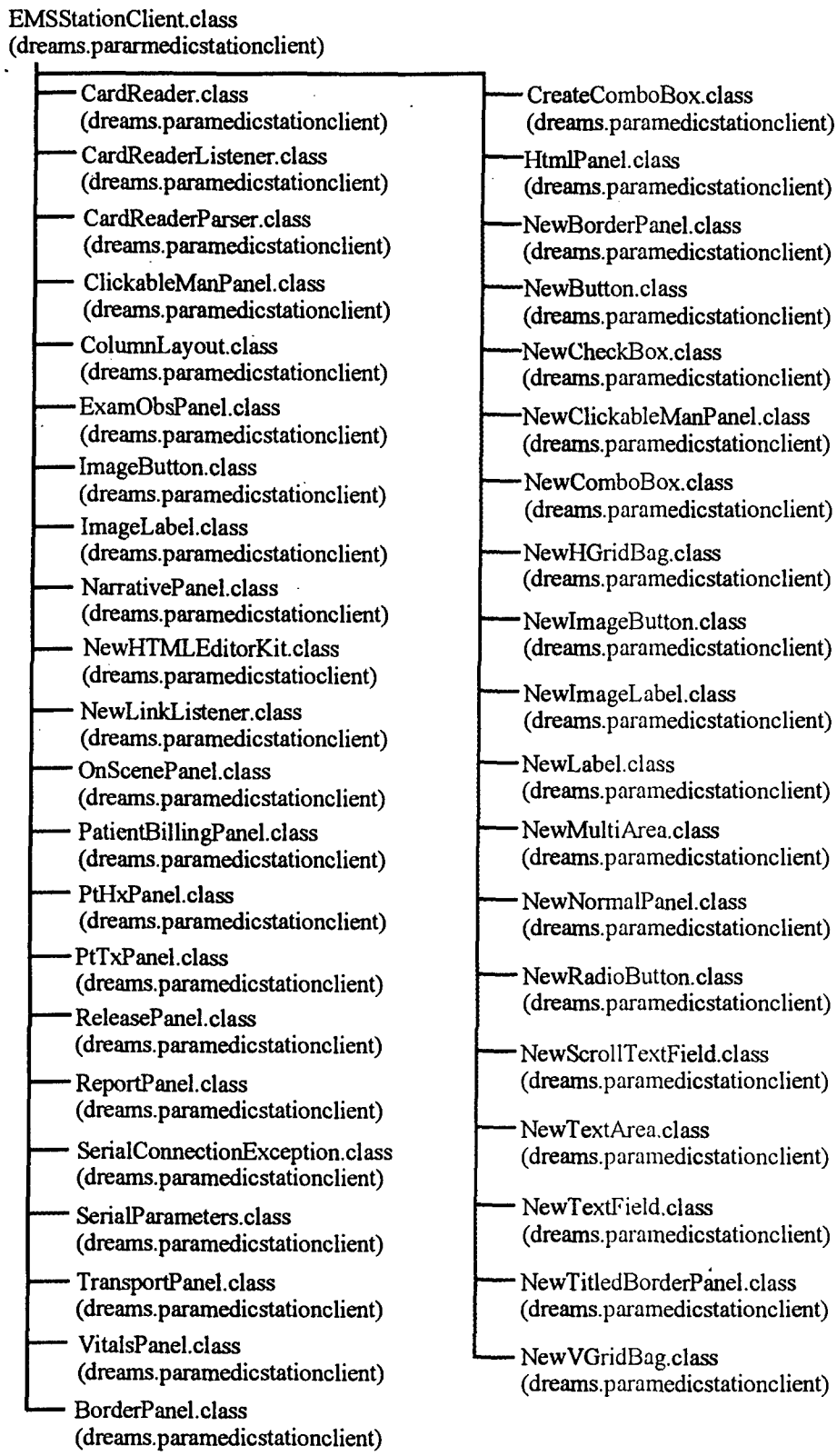


Figure A-3. Paramedic Mobile Station Client dependency graph.

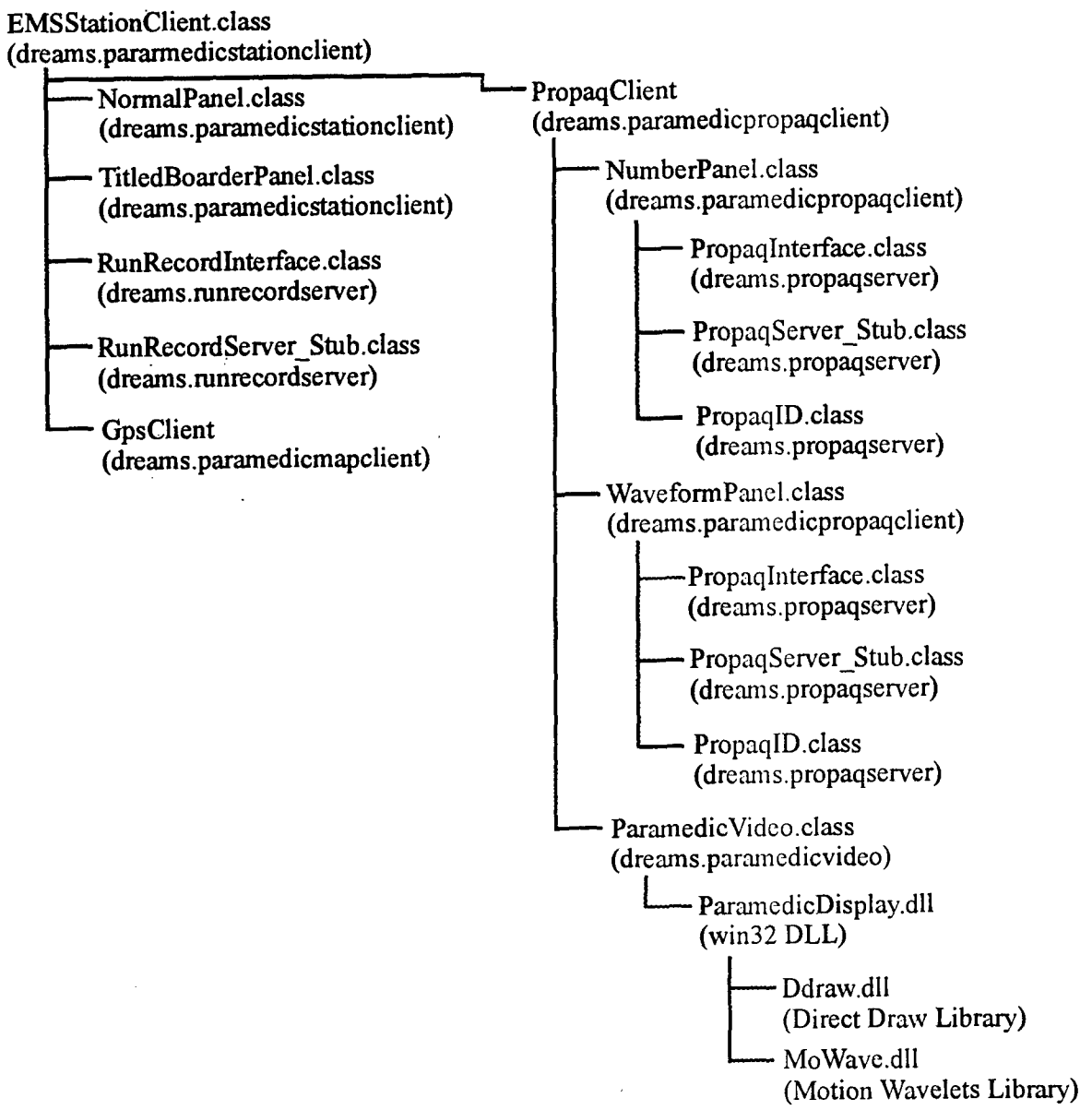


Figure A-3. Paramedic-Mobile Station Client dependency graph (continued).

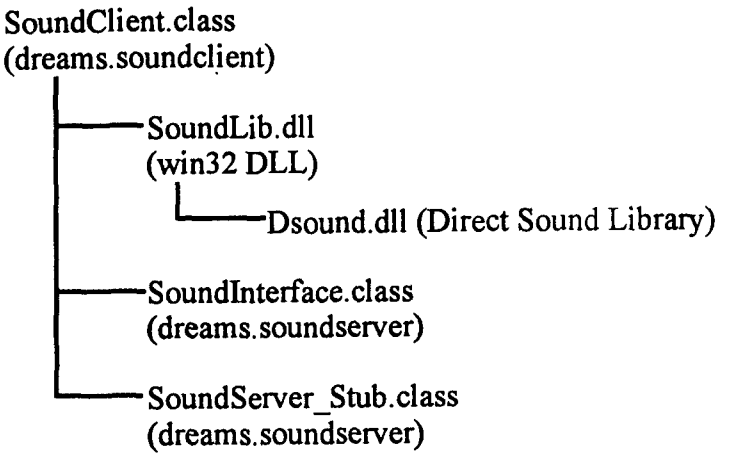


Figure.A-4. Dependency graph for Sound Client Java class.

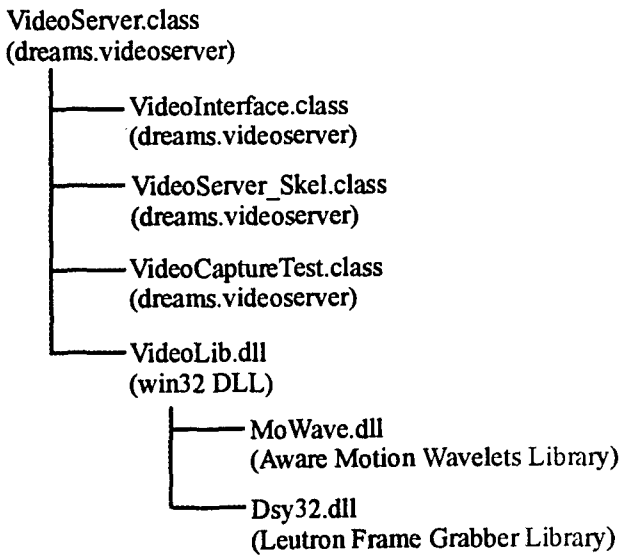


Figure A-5. Video Server Java class dependency diagram.

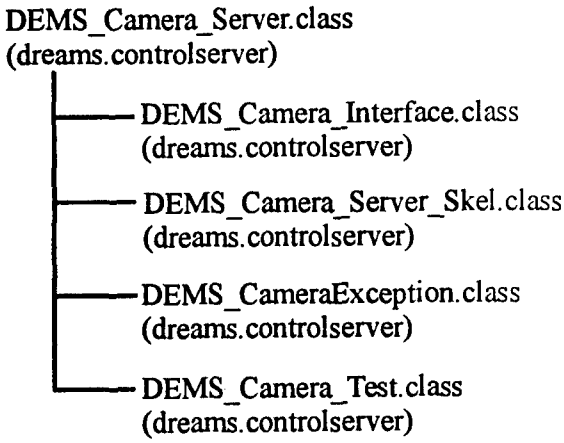


Figure A-6. Main Control Server Java class dependency diagram.

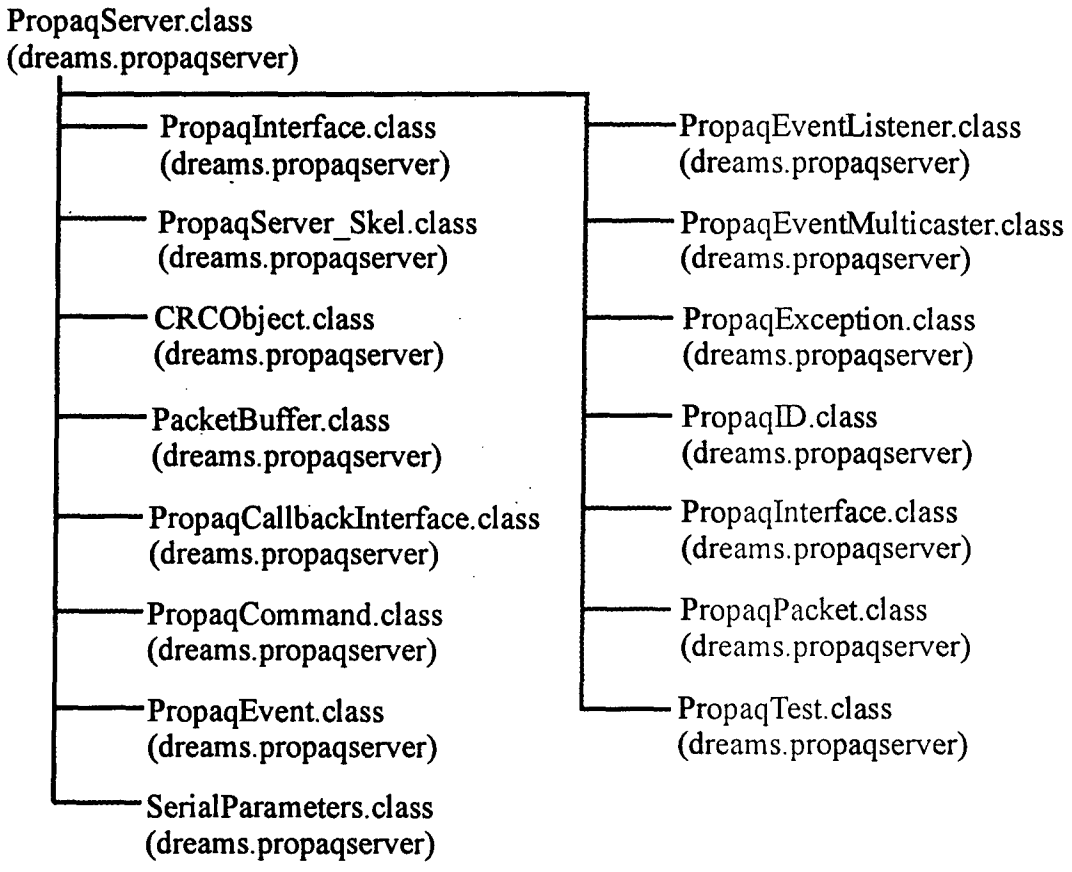


Figure A-7. Main Propaq Server Java class dependency diagram.

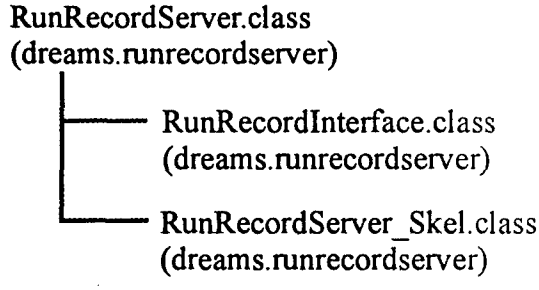


Figure A-8. Run Record Server dependency diagram.

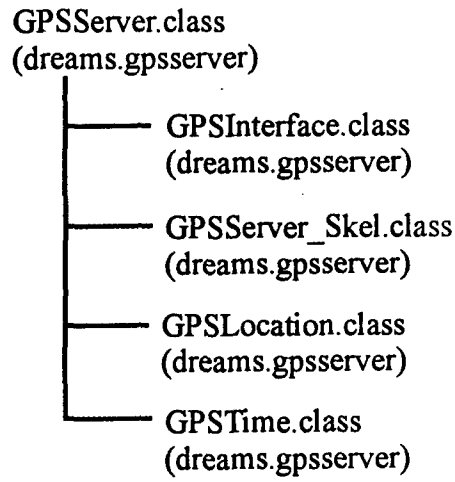


Figure A-9. GPS Server dependency diagram.

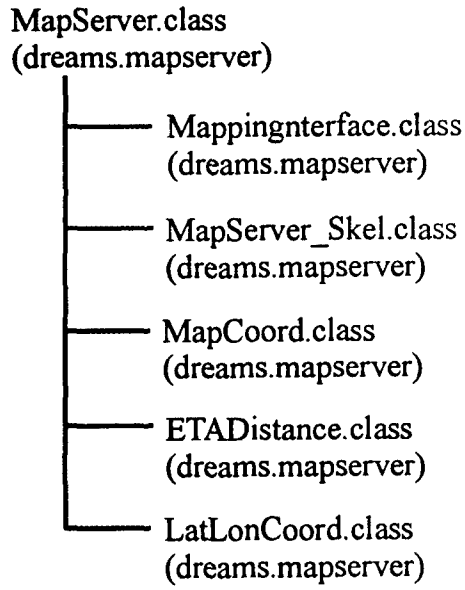


Figure A-10. Map Server dependency diagram.

Appendix B. Screen Shots of DREAMS Software

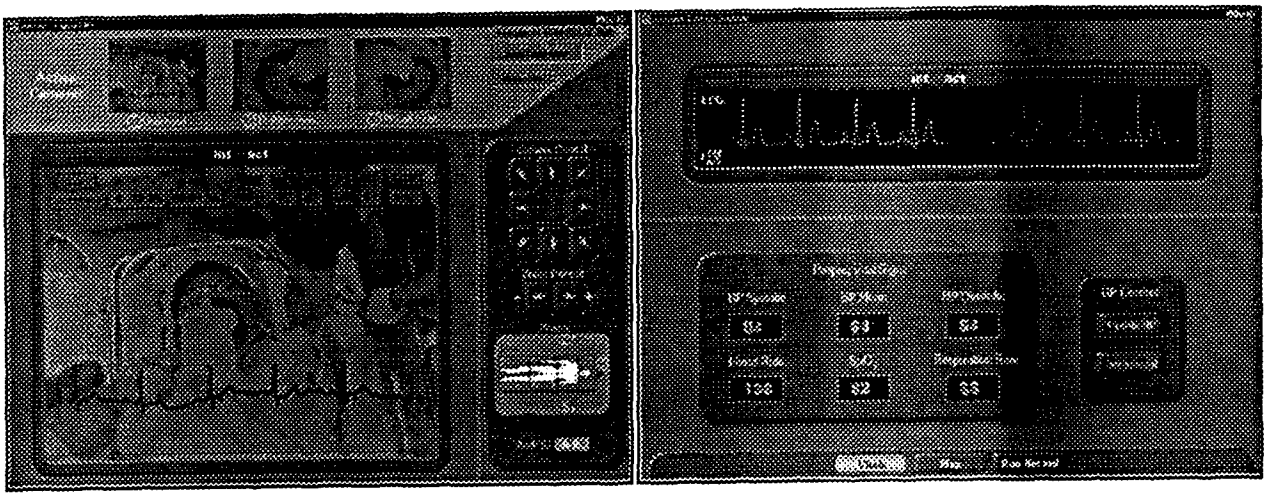


Figure B-1. Physician Video Data displays.

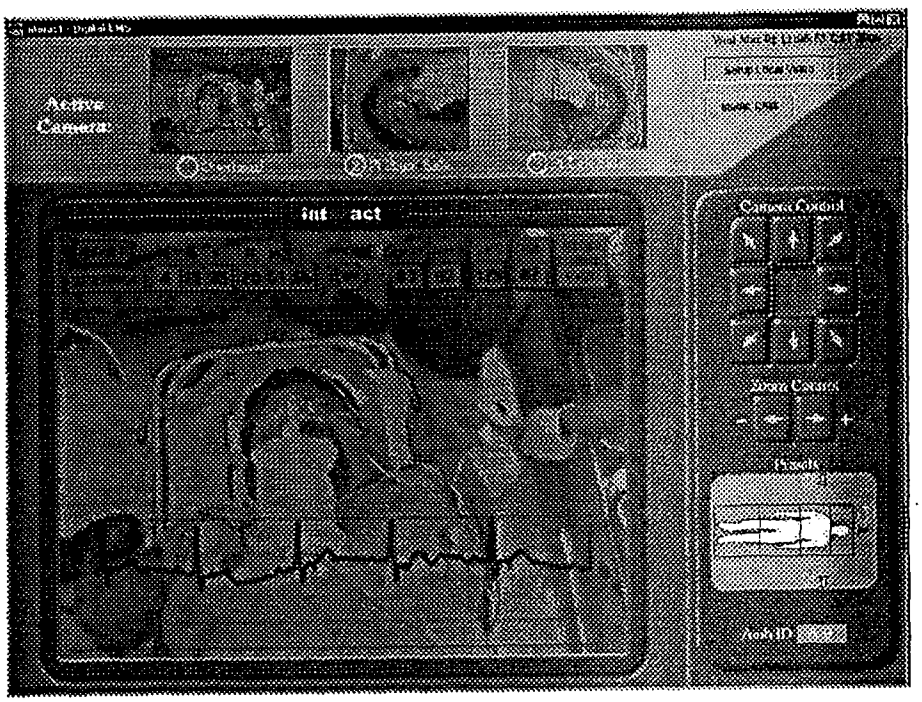


Figure B-2. Physician Video screen.

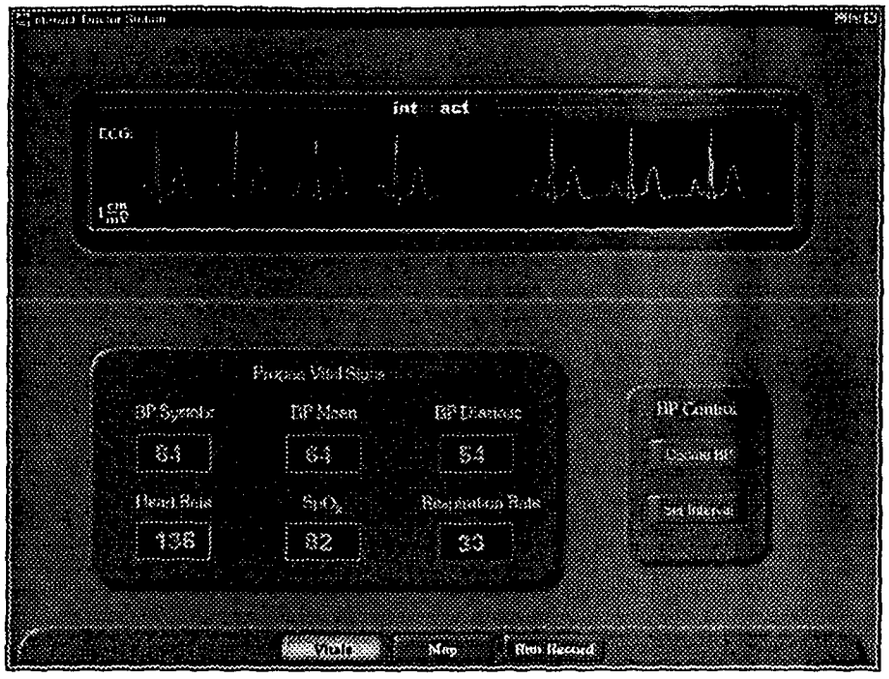


Figure B-3. Physician Vitals screen.

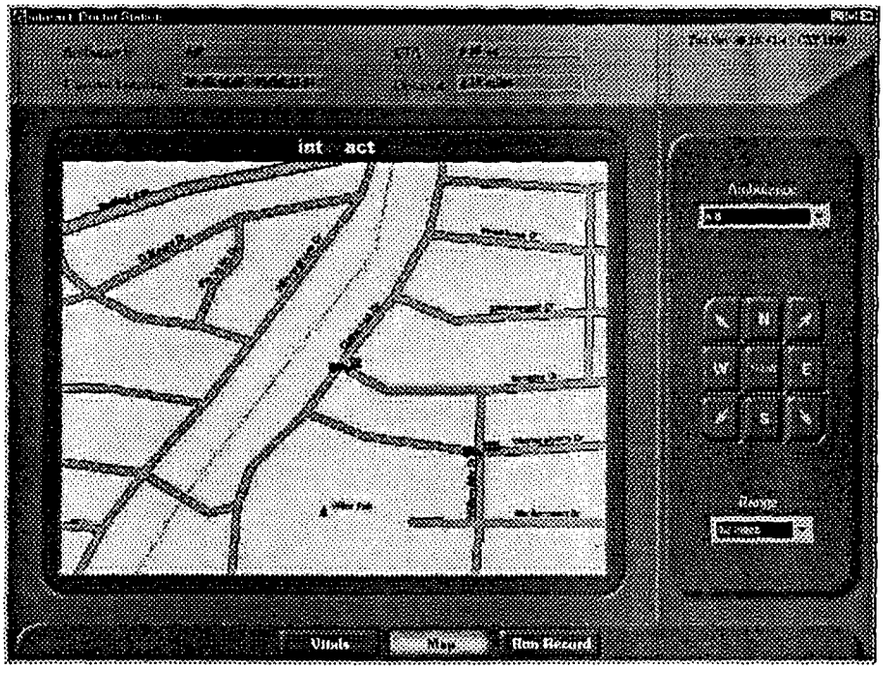


Figure B-4. Physician Map and GPS Tracking screen.

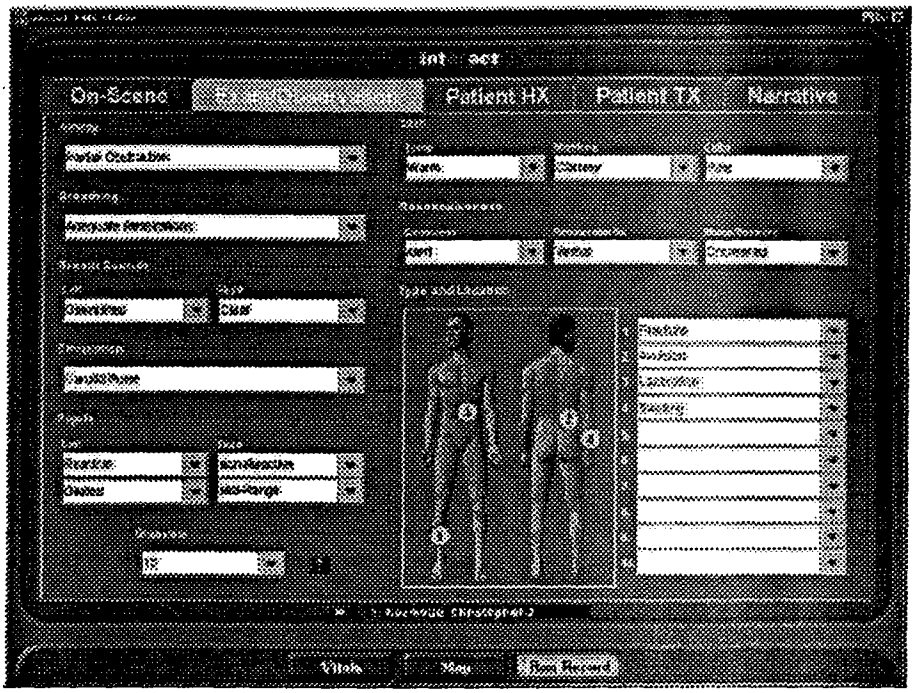


Figure B-5. Physician Run Record Examination and Observation section (Read-only).

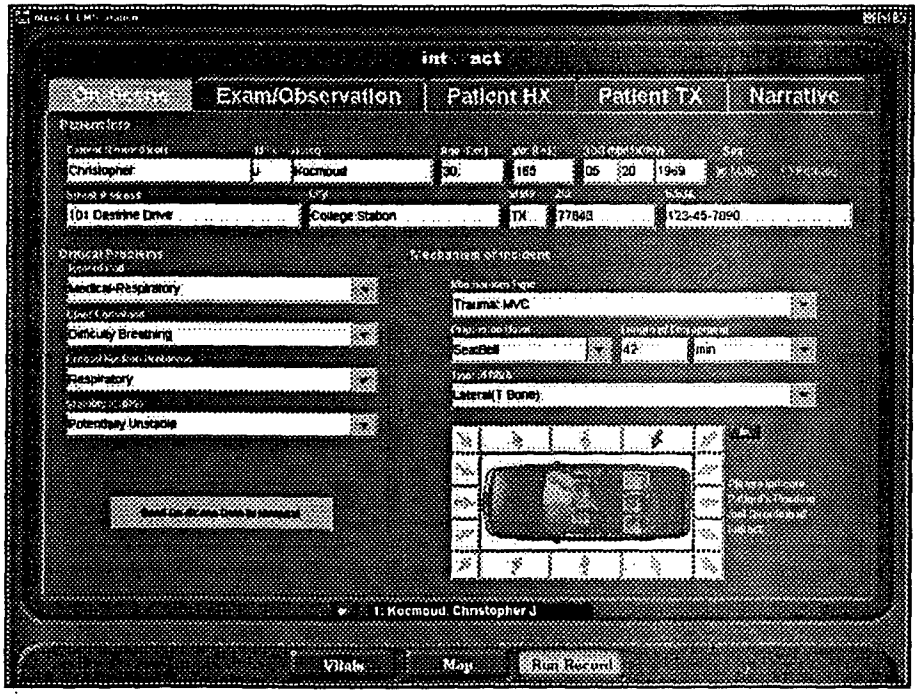


Figure B-6. Physician Run Record On-Scene section (Read-only).

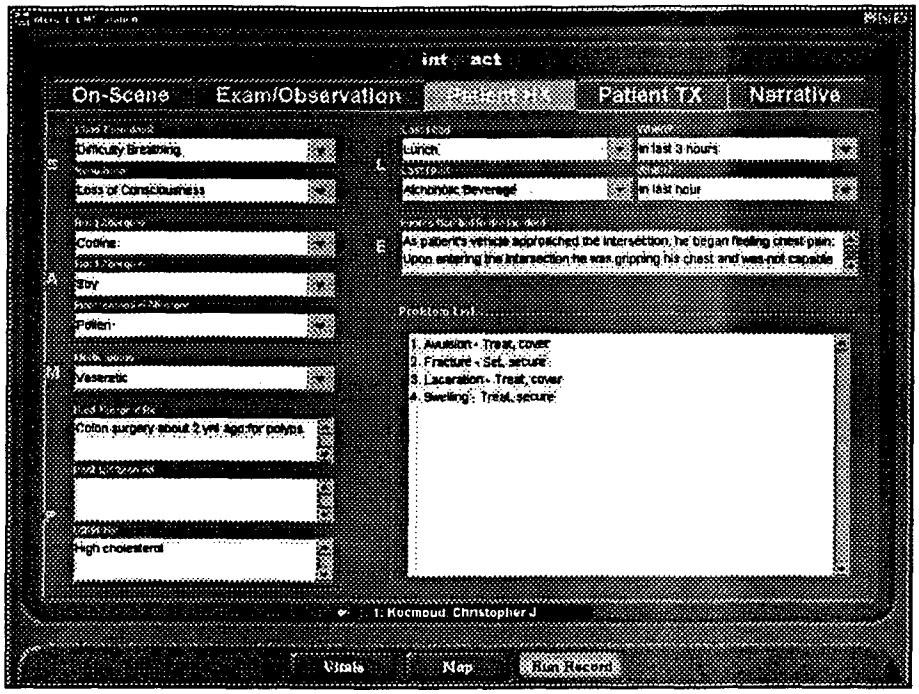


Figure B-7 Physician Run Record Patient History section (Read-only).

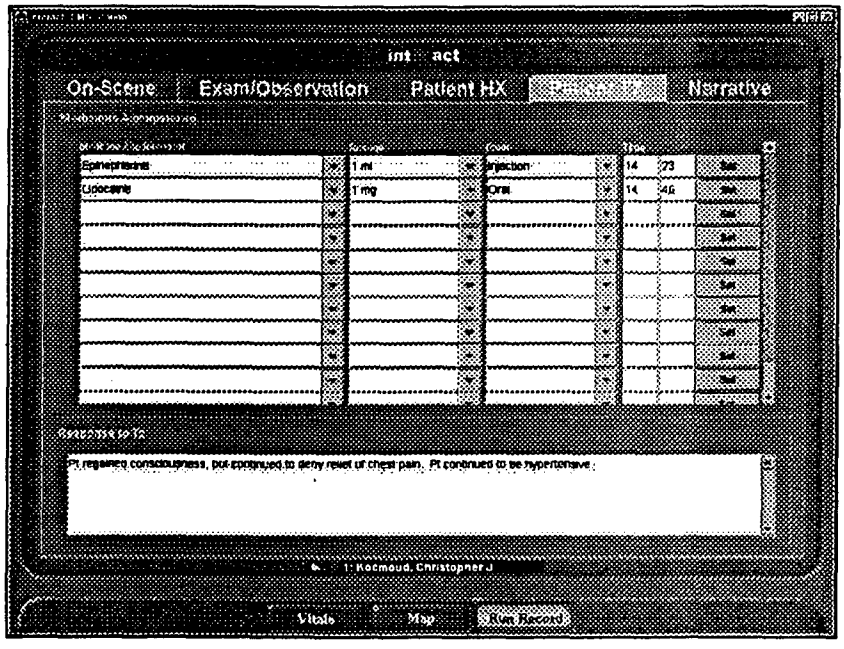


Figure B-8. Physician Run Record Patient Treatment section (Read-only).

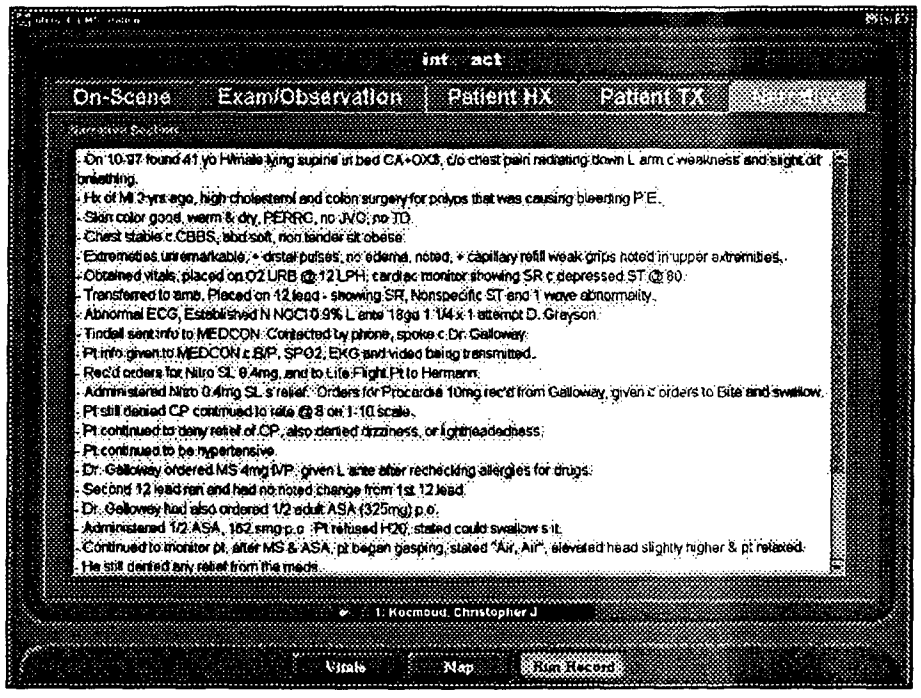


Figure B-9. Physician Run Record Narrative section (Read-only).

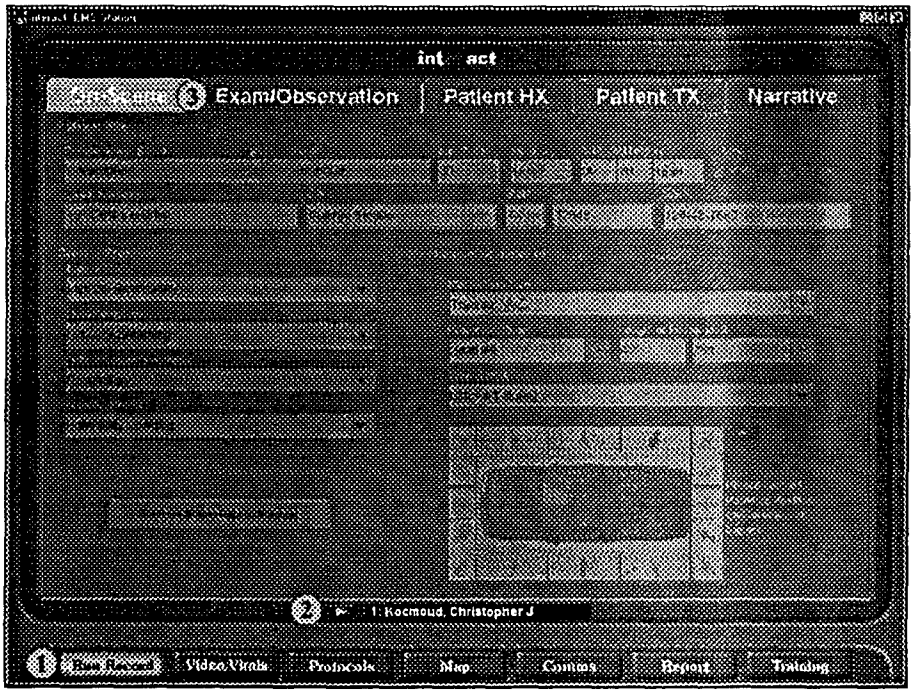


Figure B-10. Sample Paramedic-Mobile Station screen.

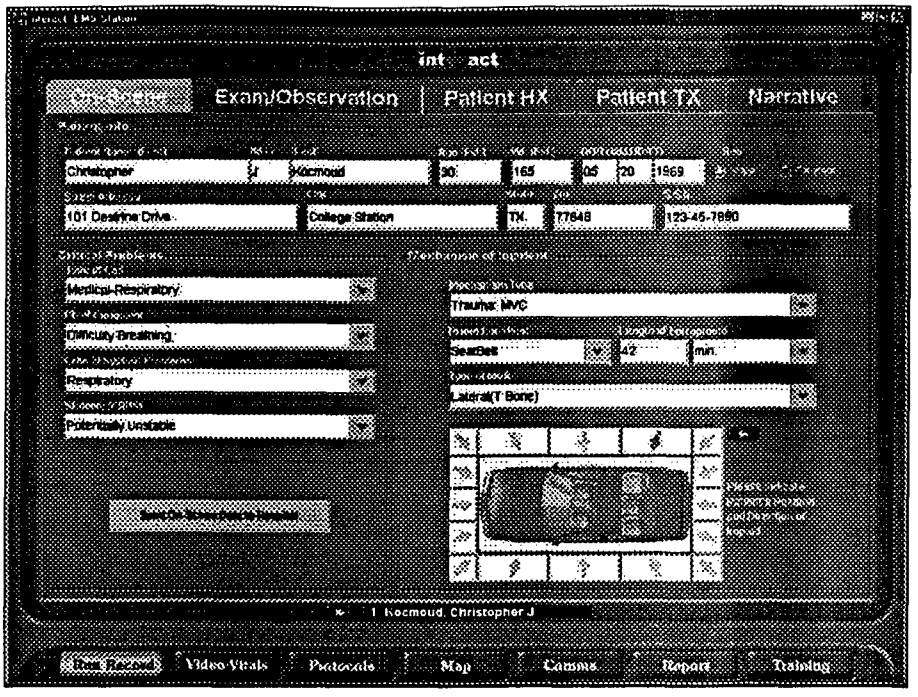


Figure B-11.Paramedic- Mobile Station Run Record On-Scene screen.

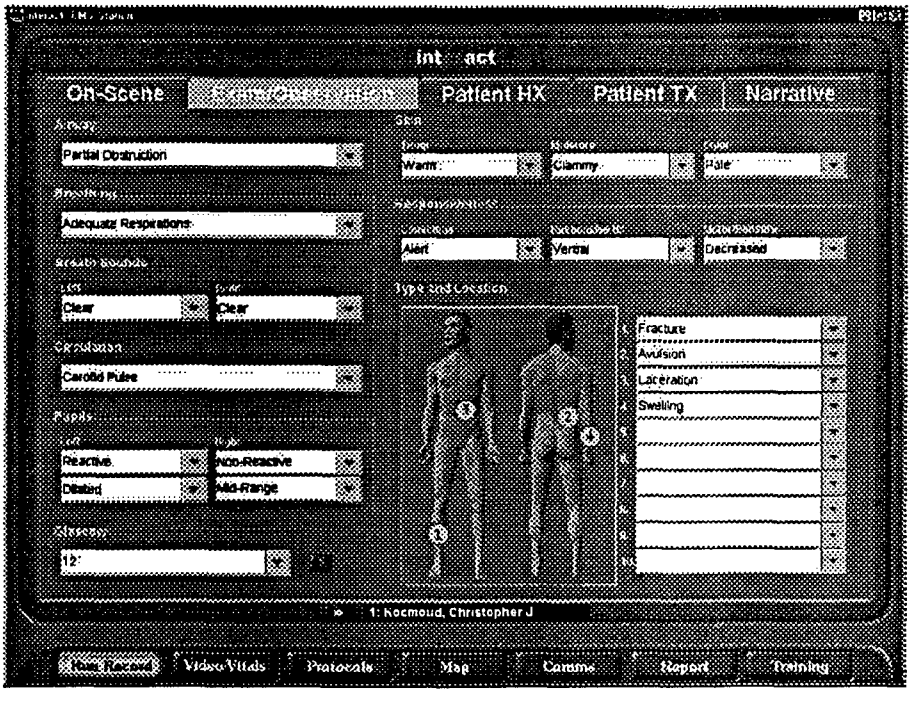


Figure B-12. Paramedic-Mobile Station Run Record Examination and Observation screen.

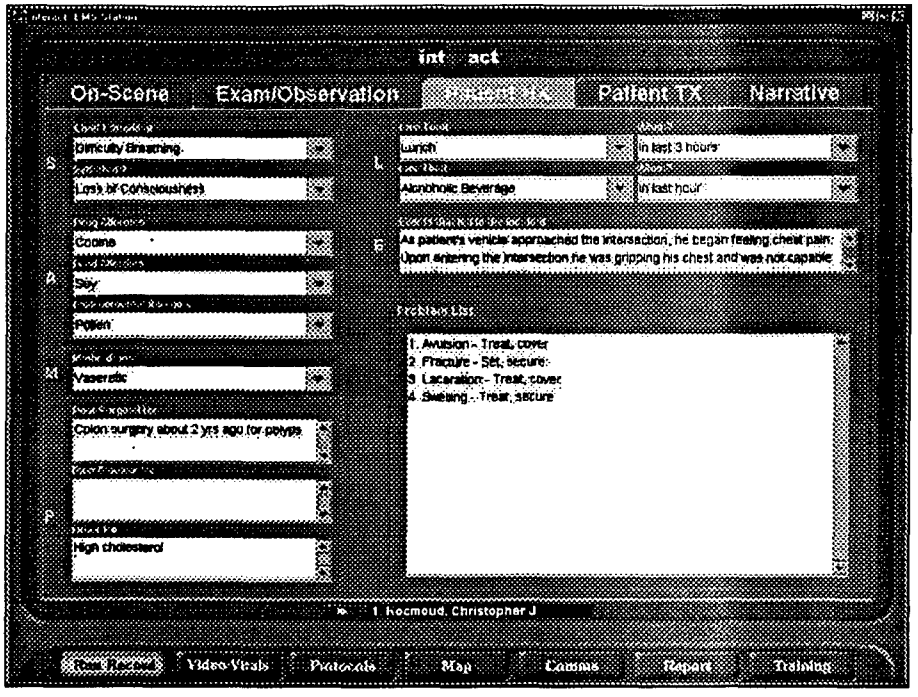


Figure B-13. Paramedic-Mobile Station Run Record Patient History screen.

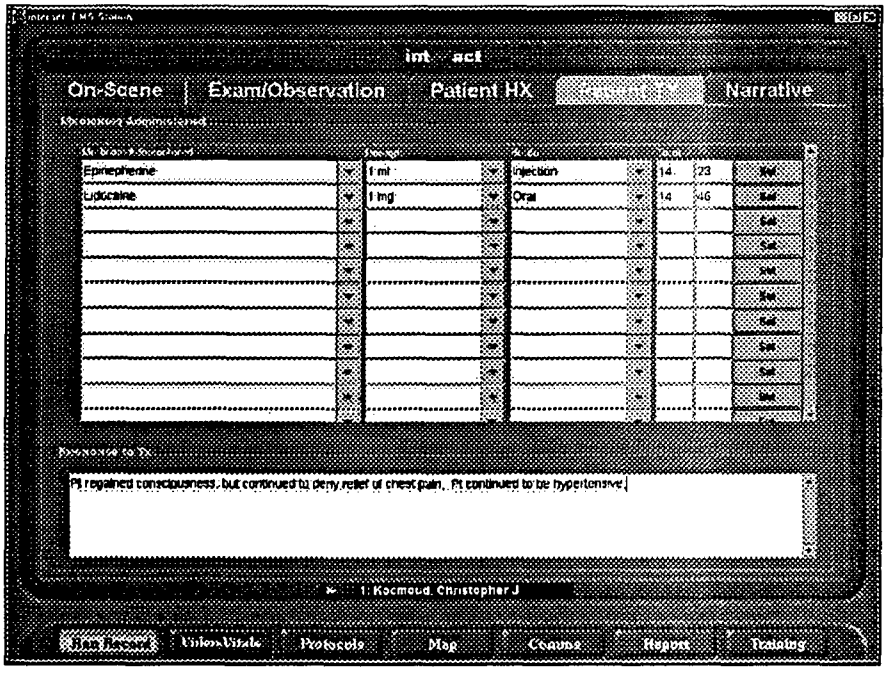


Figure B-14. Paramedic-Mobile Station Patient Treatment screen.

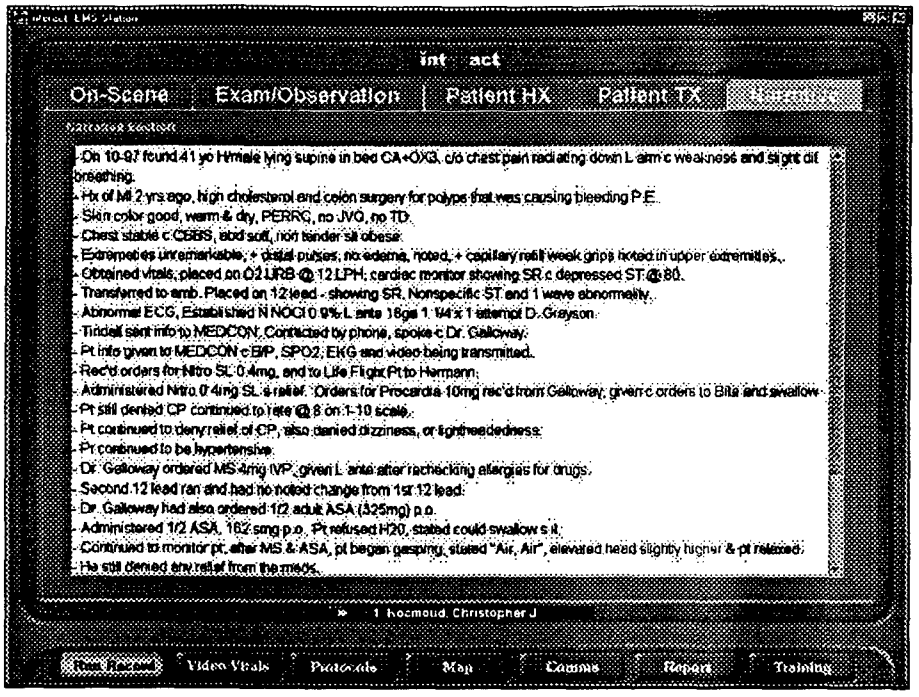


Figure B-15. Paramedic-Mobile Station Narrative screen.

Figure B-16. Paramedic-Mobile Station Video and Vitals screen.

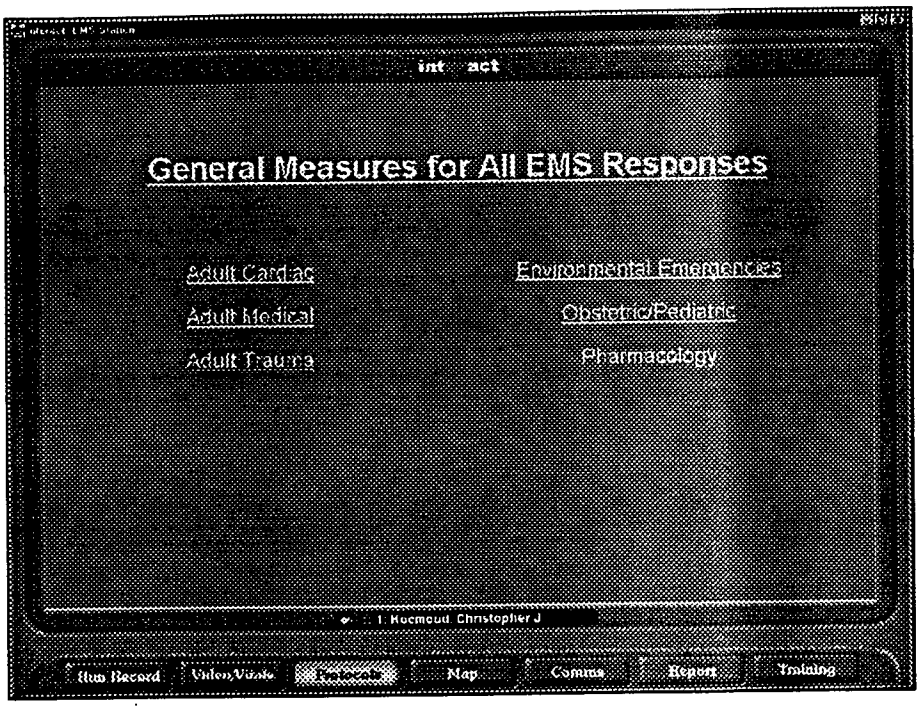


Figure B-17. Paramedic-Mobile Station On-line Protocols screen.

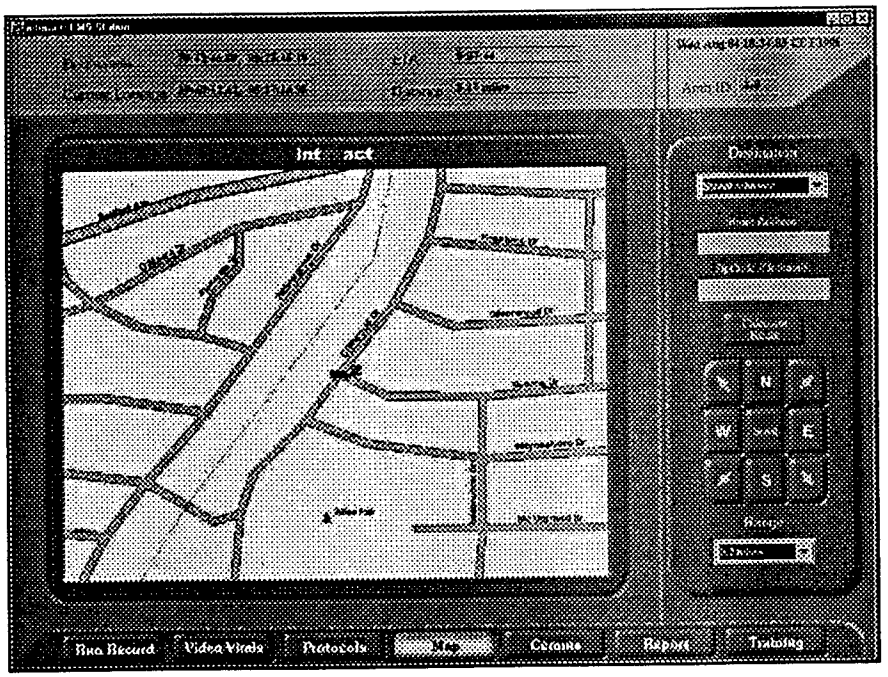


Figure B-18. Paramedic-Mobile Station Map and GPS Tracking screen.

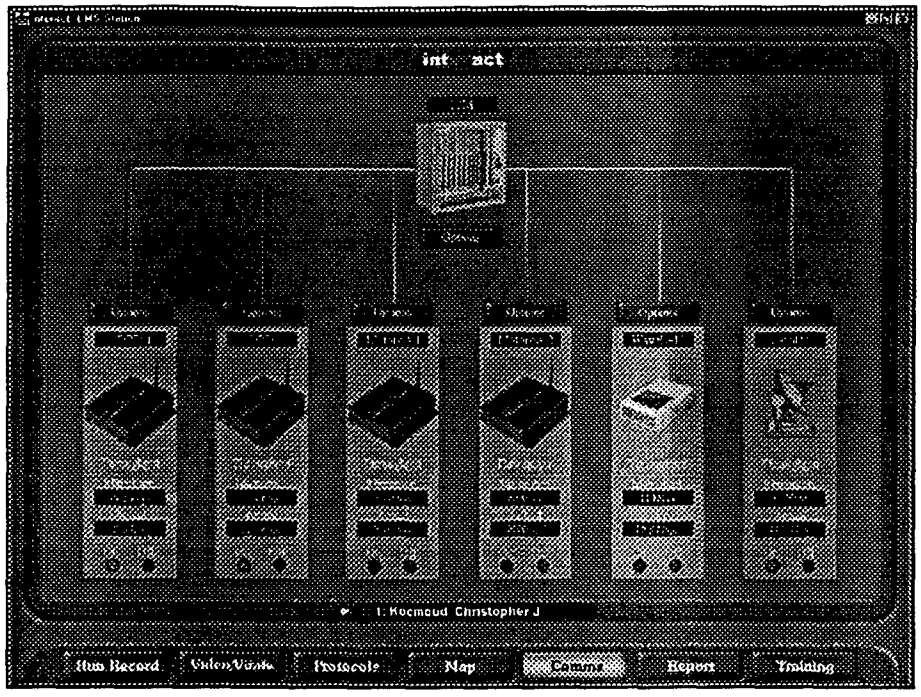


Figure B-19.Paramedic- Mobile Station Communications screen.

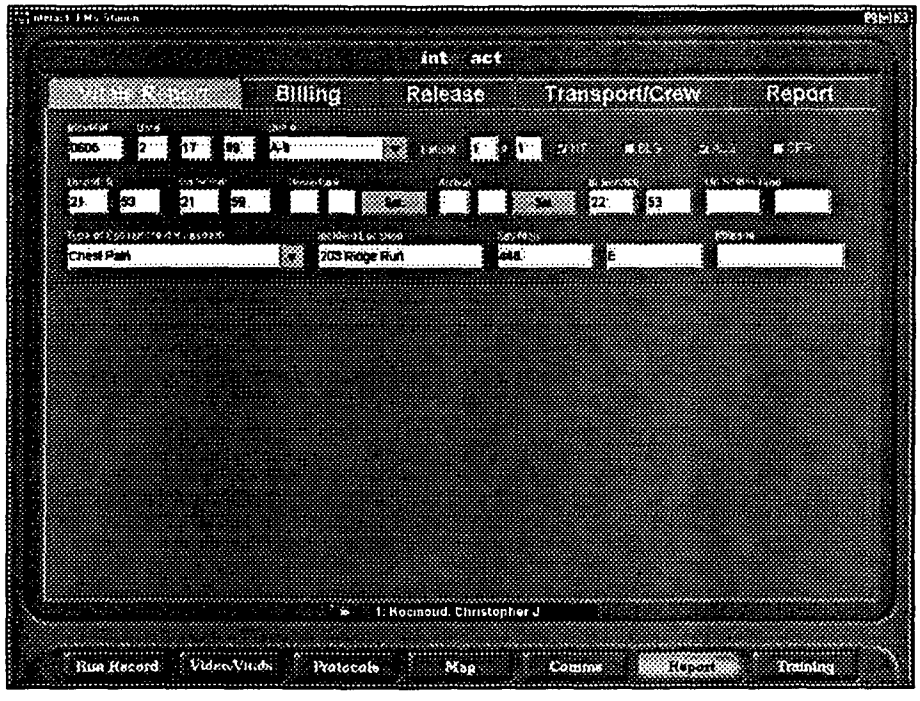


Figure B-20. Paramedic-Mobile Station Run Record and Vitals Report screen.

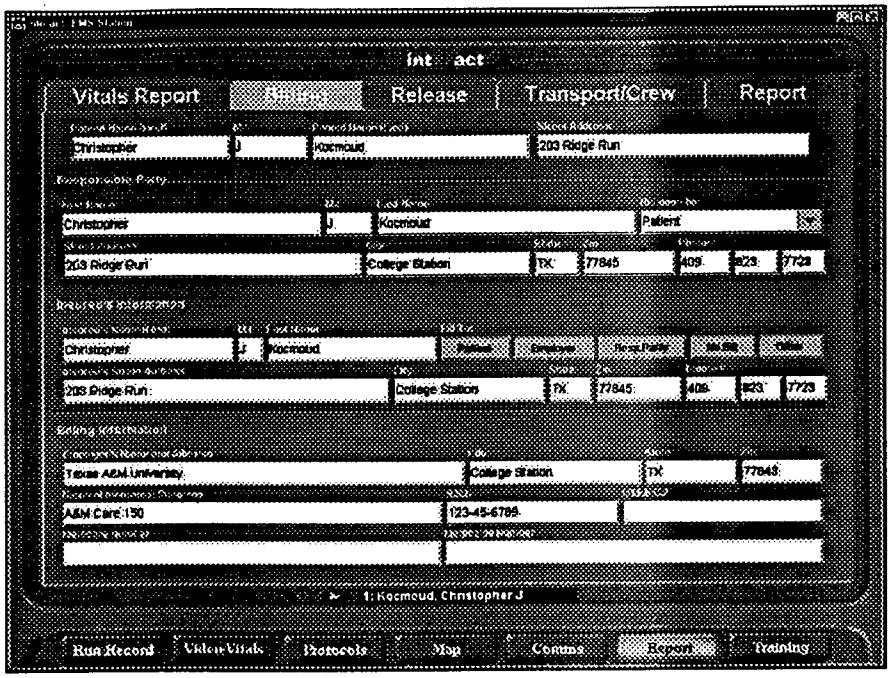


Figure B-21. Paramedic-Mobile Station Run Record Billing screen.

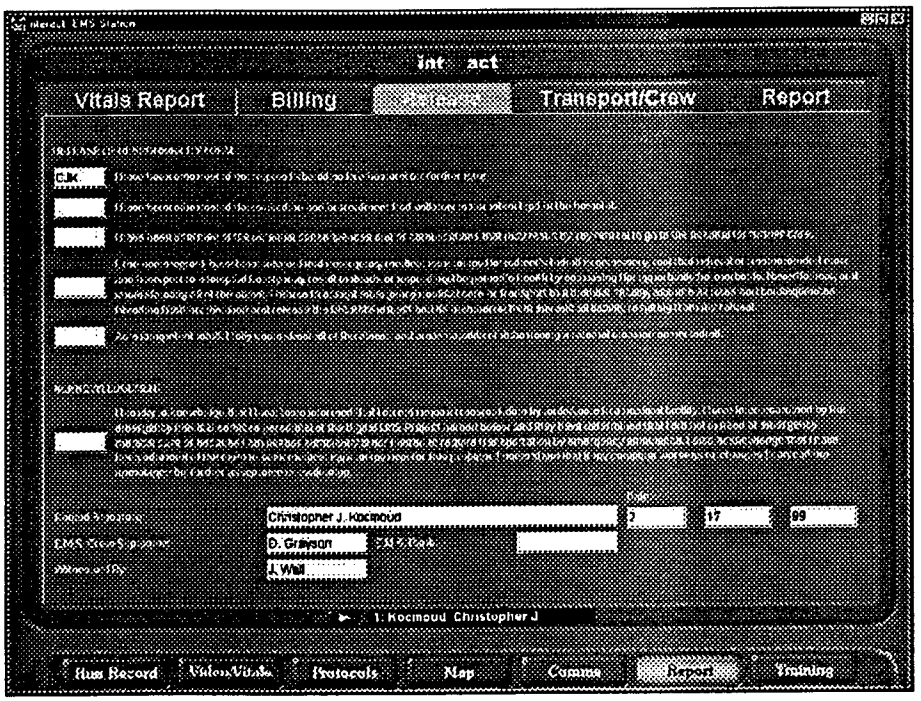
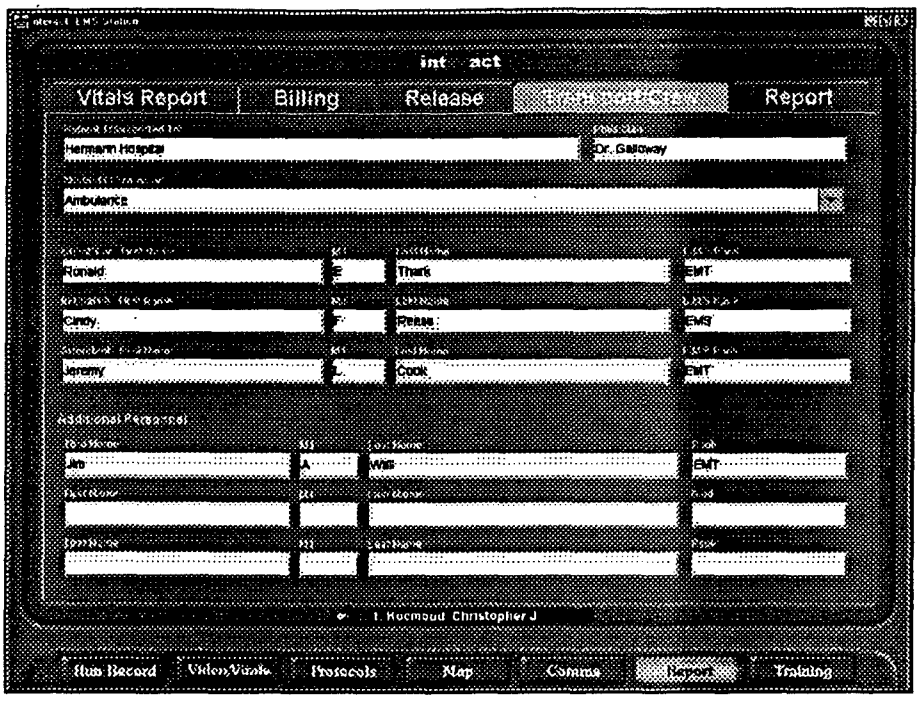


Figure B-22. Paramedic-Mobile Station Run Record Release of Responsibility and Acknowledgement screen.



Figur B-23. Paramedic- Mobile Station Run Record Transport and Crew screen.

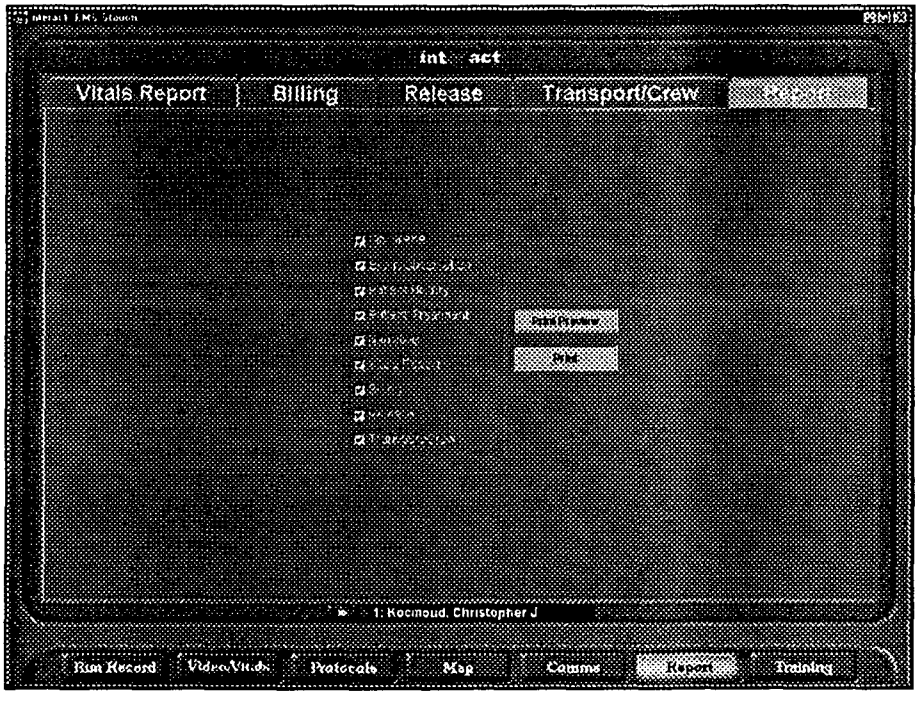


Figure B-24. Paramedic-Mobile Station Run Record Printing screen.

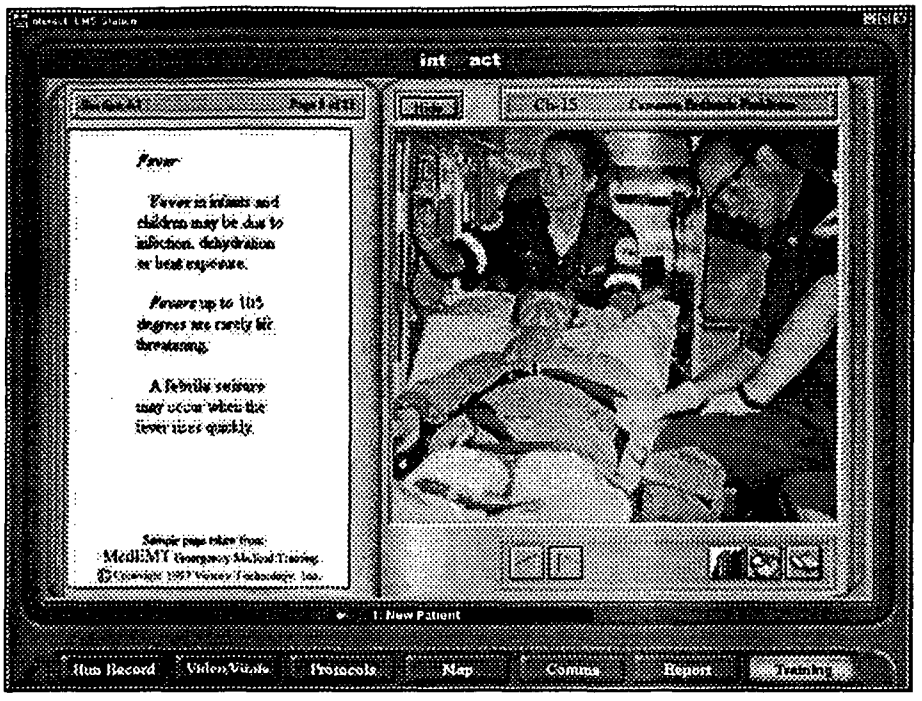


Figure B-25. Paramedic-Mobile Station Computer-Based Training screen

Appendix C. Rack Mount System Specification

PCW RM0228 Chassis

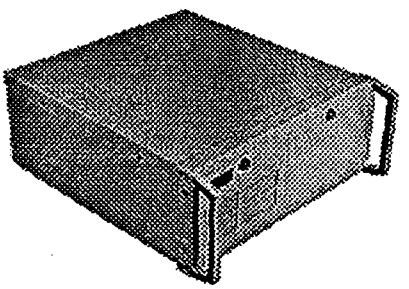


Figure C-1. PCW RM0228 Chassis.

- Suitable for 7 slots baby AT motherboard or 7 slots ATX motherboard.
- Optional 14 slot ISA,ISA/PCI, and PCI passive backplane configuration available.
- Three exposed 5.25" bays and one exposed 3.5" bay.
- Two front intake fans with replaceable air filter.
- Two locking front doors for power switch and drive bays.
- Equipped with two 120mm filtered fans.
- Rear keyboard port.
- Front handles included
- Adjustable hold-down clamps protect add-on cards against vibration

PBP14AP Backplane

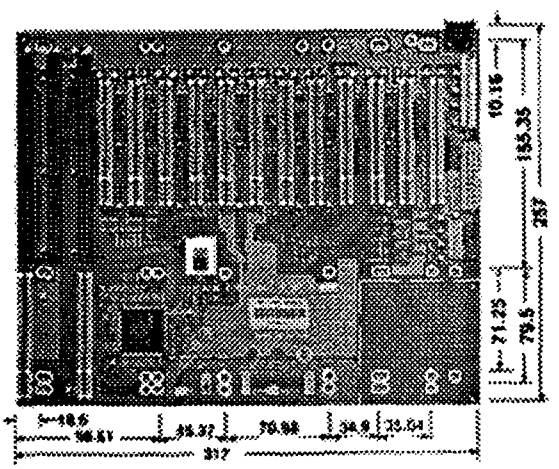


Figure C-2. PBP14AP Backplane.

- 4 Layer PCB with ground and power planes for reducing noise and keeping lower impedance Frame rated PCB at 94-V0.
- Supports ATX type and 3.3V power connector.
- LED indicator for power signals' status.

- Reserves terminal block connector for wiring.
- Equipped with gold-plated connector for better contact.
- Intel (Digital) PCI Bridge Chip.
- Support 7+ PCI Slots.
- Adopts Tantalum capacitors to keep circuit signal more reliable and stable.
- 2 ISA Bus Slots, 11 PCI Bus Slots.
- 1 PICMG Cpu Card Slots (2 Different Slots Locations).
- Digital PCI-PCI Bridge Chip.
- Support AT or ATX Power Source.
- Fits Standard 20 Slots Rackmount Chassis.

SBC-738 Single Board Computer

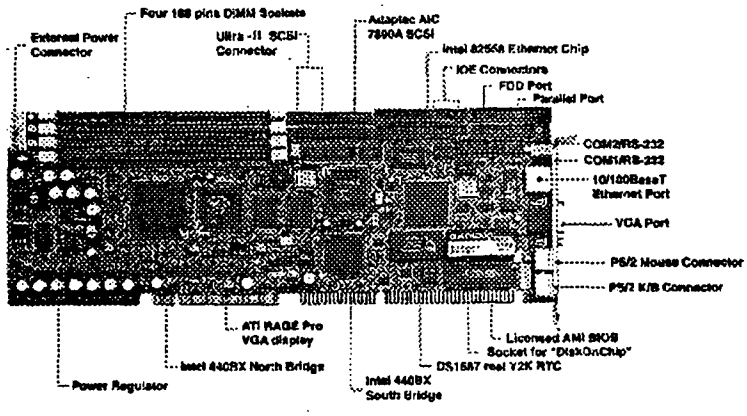


Figure C-3. SBC-738.

SBC-738 specs:

- Full functionality full-size all-in-one high speed Pentium II SBC.
- Supports Intel Pentium II Slot 1 CPU upto 500Mhz (66/100Mhz Bus clock).
- Complied with PC98, PCI V2.1 and PICMG 2.0 standard.
- Fastest 80MB/s Disk I/O on-board (Ultra-II SCSI).
- On-board ATI Rage Pro AGP graphic port.
- 10/100 BaseT Ethernet port.
- On-board Watchdog Timer.
- On-board ATX power control.
- DiskOnChip socket supporting memory size up to 144MB.
- Super Fast 100Mhz Bus Speed PICMG solution for computer telephony applications.
- CPU: Supports Intel Pentium II Slot 1 CPU up to 500Mhz (66/100Mhz Bus clock) (Not Included).
- BIOS: Flash (AMI).
- Chipset: Intel 80440BX AGP Chipset.
- EIDE: Two Enhance IDE Ports (Supports Ultra DMA 33).
- FD: One Floppy Port.
- Printer Port: One ECP/EPP bi-directional parallel printer port.
- Video: ATI Rage Pro Chip, supports 2/3D graphics with 4MB DRAM.
- Network: Intel 82558 Ethernet chip.

- Network: One 10/100 BaseT Ethernet port.
- SCSI: Adaptec AIC 7890A SCSI Controller.
- SCSI: Supports two 68 pins Ultra-2 SCSI ports, up to 80MB/s transfer rate.
- System memory: Up to 1 GB SDRAM.
- Memory module: SDRAM × 4.
- Solid State Disk: Optional Flash disk (DiskOnChip) up to 144MB.
- Other features: Two serial ports (Winbond W83977ATF Chip), two USB ports.
- Supports: keyboard cable for passive backplane board.
- Supports: One PS2 Mouse and PS2 Keyboard connectors.
- Watchdog Timer: Programmable 0.5, 1,2,4,8,16,32,64 sec time-out interval.
- Suitable for PICMG PCI/ISA passive backplane board.
- Power Requirements: +5V@10.0A (typical), +12V@80mA, -12V@20mA.
- Operating Temperature:0 to 60 degree C.
- Relative Humidity:5 to 95% non-condensing.
- Bus: PICMG (PCI/ISA Bus).

Power Supply

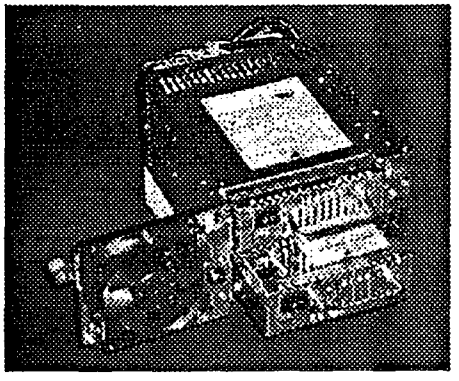


Figure C-4 Power Supply.

Power supply specs:

- Compact size, removable, door mounted, with fan.
- Load balance sharing design for power redundancy.
- Excellent ventilation design for durability.
- Compatible with Windows NT UPS Service through FAB-5 (For message broadcast, shut down operation).
- Remote sensing design for better regulation.
- Provides hot-plug, hot-swap ability.
- With DC fan failure detection function.
- Color-changing warning LED, alarm buzzer, Power Faulty signal delivery once defect occurs.
- ATX 2.0 Features (720mA 5VSB, 3.3V DC Output, Remote On/Off) are optional.
- Individual power unity On/Off switch securely located.
- Meets UL 1950, CSA 22.2 No. 234, TUV IEC 950.
- Meets FCC Class B, CISPR 22 Class B.
- Auto cut-off of power for fan upon opening of revolving door.

- Rail design for power unit allows smooth sliding in/out.
- Thumb screw design for easy maintenance.
- Input Characteristics: (RP5250AT / RP6250ATX).
- Voltage: 98-132 VAC/190-260 VAC switchable.
- Frequency: 47-63 Hz.
- Input Current: 3.0 A/ 1.5 A per unity (parallel), 6.0 A/3.0 A (One power).
- Inrush Current: 30 A Max. for 115 VAC, 60 A MAX, for 220 VAC.

Ruggedized Hard Disks

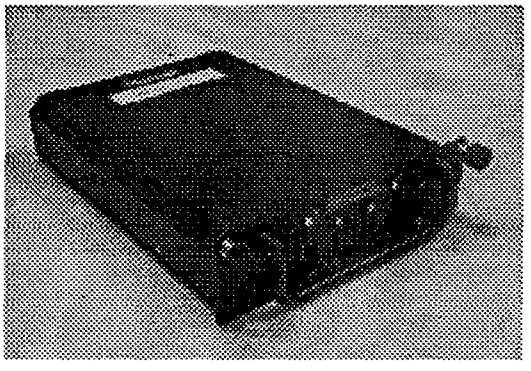


Figure C-5. Moutain Optec ruggedized hard disk.

Mountain Optech's 5.25" FORM FACTOR HARD DRIVES - Capacities from 9GB to 18GB, some with removable drive modules. This enables virtually unlimited capacity, with off-line storage for mission critical and sensitive data. Any of MOI's drives can be custom packaged or mounted into 19" rack-mount units. The removable disk drive module can be placed in a 19", 5.25" or CS type enclosure. Conformal coating of electronics provides moisture resistance and reduces susceptibility to salt-fog and other environmental contaminants. Built-in fans direct airflow over the electronics, while temperature firmware and heaters provide for a wide operating temperature range. See Figure C-6 for mechanical specifications.

MOUNTAIN OPTEC HARD DISK SPECIFICATIONS	
Dimension	5.25"
Transfer Rate	23.7 Mbps
Average Seek time	9.5 msec
Average Latency	4.17 msec
Rotational Speed	7200 rpms
Corrected read BER	10^{-14}
Drive Interface	SCSI
Capacity	9 GB
Temperature	-15 to +55 C Operational
	-40 to 70 Survive
Shock op / survival	20G / 30G
Vibrations (20-2KHz)	2grms/ 4grms
Humidity	5 to 95%

Attitude	Any
Altitude (feet)	-1K to 20K
Dimensions wxhxd	4x3.25x5.75
Weight	5lbs
Power	12 Watts

Figure C-6. Mountain Optec Hard Disk Specifications.

Appendix D. Communications Equipment Specification.

Sierra Wireless MP200 CDPD Modem

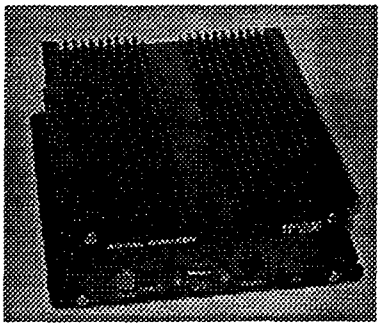


Figure D-1. Sierra Wireless MP200 CDPD Modem.

Technical Specifications:

- Dimensions (inches): 7.25L x 7W x 2.25H.
- Power Supplies:
- Standard 13.8V vehicle battery.
- RF Output.
- Class 1 mobile device for AMPS and CDPD operation.
- Up to 3 watts power output from device (Higher ERP depending on antenna & cabling).
- Antenna Interface.
- 50 ohm RF connector.
- Host Interface.
- DB-9 with RS-232 signal levels with speeds up to 57.6kbps.
- Standard Modem/Host Software Interface.
- AT command set (Circuit Switch).
- SLIP (CDPD).
- Transmitter Disable.
- CDPD Transmitter may be selectively disabled to prevent transmission in sensitive areas.
- Data Protocols.
- CDPD Release 1.1.
- Optional Circuit Switched Fax Compatibility.
- CCITT V.17 14.4kbps fax.
- CCITT Group III Class 1 fax.
- Optional Circuit Switched data communications.
- CCITT V.42, CCITT V.42bis, CCITT V.32bis, CCITT V.22bis.
- Enhanced Throughput Cellular (ETC) Protocol.
- Environmental Limits.
- Operating Temperature.
- -22F (-30C) to 158F (+70C).
- (CDPD Mode, restricted duty cycle).
- -22F (-30C) to 140F (+60C).

- (CDPD Mode, unrestricted duty cycle).
- -22F (-30C) to 140F (+60C).
- (AMPS Mode option, restricted duty cycle).
- Storage Temperature -40F (-40C) to 176F (+80C).
- Vibration MIL-STD 202F.
- Humidity 5 to 95% non-condensing.

CDPD Antenna

- Cellular Hard Mount Antenna (PN6000065) Mobile Antenna.
- High Gain (3dB) antenna with low loss cable (0.5 dB/m).

BreezeCom IEEE 802.11

BreezeNET AP-DS.11 Base Station

The AP-DS.11 11 Mbps base station (network access point) is used to create a wireless extension to a wired network or can be used to create a completely wireless network thus eliminate the need for costly, time-consuming Ethernet cable installation. The Access Point has a RJ-45 connector which can be easily connected to an Ethernet hub or to a 10BASE-T connection on a server, creating a wireless cell in which DS.11 client devices (PC Cards, Station Adapters, and ISA Adapters) can roam. Once the AP-DS.11 is plugged in and connected to the network, configuration of both wireless and wired network components can be easily accomplished using an SNMP management workstation. The AP-DS.11 also functions as a bridge (does not transmit peer to peer traffic). This item is shown in Figure D-2.

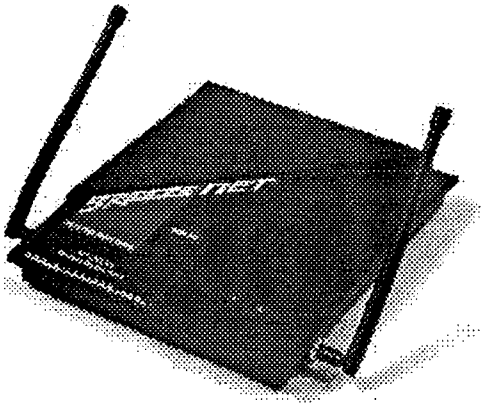


Figure D-2. AP-DS.11 BreezeNET Base Station.

The AP-DS.11 Access Point uses direct sequence spread spectrum radio technology operating in the 2.4 GHz industrial, science, medical (ISM) band to transmit data at speeds of up to 11 Mbps. It is fully compatible with the IEEE 802.11 TGb wireless LAN standard and transparently supports all popular Ethernet protocols. It is also compatible with 802.11-compliant, 2 Mbps direct sequence clients.

BREEZENET DS.11 SPECIFICATIONS	
Product Features	**AP, SA,WBS,WBC**
Data Rate	1, 2, 5.5, 11 Mbps
Aggregate Throughput	33 Mbps (3 Co located links)
Office environment	150 ft
Open Space indoors	1000 ft
Open Space Outdoors	15 Miles
Max No. of Co-located channels	3
Max No of channels	USA 11
Load Sharing	Yes
Compliant with	Ethernet/Ieee 802.3 CSMA/CD 802.10 VLAN
Physical interface	RJ45 10BaseT
Network operating system	All
Network drivers	N/A
Type	Direct Sequencing Spread Spectrum
Wireless LAN Standard	IEEE 802.11
Frequency range	2.4 – 2.4835 GHz
Sensitivity @ 1 Mbps	-89dBm 1E-5
Sensitivity @ 2 Mbps	-86dBm 1E-5
Sensitivity @ 5.5 Mbps	-84dBm 1E-5
Sensitivity @ 11 Mbps	-80dBm 1E-5
Modulation @ 1 Mbps	BPSK
Modulation @ 2 Mbps	DQPSK
Modulation @ 5.5 Mbps	CCK
Modulation @ 11 Mbps	CCK
Power USA	+18 dBm
Europe	18 dBm
Approvals of compliance	FCC part 15, ETS 300-328
RF Connectors	Proprietary SMA
Configuration and setup	Windows DS.11 Manager Utility, SNMP
Indicators	Yes, 3 Leds
SNMP	Yes
Input Power Supply	100 – 240VAC
Output Power Supply	9 vdc 1A
Safety compliance	UL/ULC ,TUV/GS, CE
Dimensions	8x9.5x1.2 inches
Weight	3.25 pounds
Operating Temperature	0C to 40C
Operating Humidity	10% to 90%

Figure D-3. BreezeNET DS.11 Specifications.

BreezeNET Wireless Bridge WBC-DS.11

The WBC-DS.11 is a wireless bridge that connects a remote (mobile) Ethernet network to a WBC-DS.11 Multipoint Base Station bridge located at a central server or base station. It can handle up to 1024 MAC addresses and can achieve links of up to 15 miles.

The WBC-DS.11 Remote Workgroup Bridge uses direct sequence spread spectrum radio technology operating in the 2.4 GHz industrial, science, medical (ISM) band to transmit data at speeds up to 11 Mbps. It is fully compatible with the IEEE 802.11 TGb wireless LAN standard and transparently supports all popular Ethernet protocols.

BREEZENET WBC-DS.11 SPECIFICATIONS	
Product Features	**AP, SA,WBS,WBC**
Data Rate	1, 2, 5.5, 11 Mbps
Aggregate Throughput	33 Mbps (3 Co located links)
Office environment	150 ft
Open Space indoors	1000 ft
Open Space Outdoors	15 Miles
Max No. of Co-located channels	3
Max No of channels	USA 11
Load Sharing	Yes
Compliant with	Ethernet/Ieee 802.3 CSMA/CD 802.10 VLAN
Physical interface	RJ45 10BaseT
Network Operating system	All
Network drivers	N/A
Type	Direct Sequencing Spread Spectrum
Wireless LAN Standard	IEEE 802.11
Frequency range	2.4 – 2.4835 GHz
Sensitivity @ 1 Mbps	-89dBm 1E-5
Sensitivity @ 2 Mbps	-86dBm 1E-5
Sensitivity @ 5.5 Mbps	-84dBm 1E-5
Sensitivity @ 11 Mbps	-80dBm 1E-5
Modulation @ 1 Mbps	BPSK
Modulation @ 2 Mbps	DQPSK
Modulation @ 5.5 Mbps	CCK
Modulation @ 11 Mbps	CCK
Power USA	+18 dBm
Europe	18 dBm
Approvals of compliance	FCC part 15, ETS 300-328
RF Connectors	Proprietary SMA
Configuration and setup	Windows DS.11 Manager Utility, SNMP
Indicators	Yes, 3 Leds
SNMP	Yes
Input Power Supply	100 – 240VAC
Output Power Supply	9 vdc 1A
Safety compliance	UL/ULC ,TUV/GS, CE
Dimensions	8x9.5x1.2 inches
Weight	3.25 pounds
Operating Temperature	0C to 40C
Operating Humidity	10% to 90%

Figure D-4. BreezeNET WBC-DS.11 Specifications.

Base Station Omni Directional Antenna

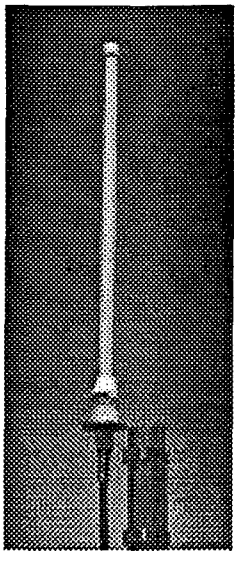


Figure D-5. Base Station OMNI-8 omni directional antenna.

The OMNI-8, 8-dBi, omnidirectional antenna can be used with the BreezeNet PRO.11 D Model Bridges and DS.11 Outdoor Bridges. It is used for outdoor, multipoint links that need 360 degree dispersion pattern.

The OMNI-8 antenna attaches to the female connector on the PRO.11 D Model Bridges and the DS.11 Outdoor Bridges and provides 360 degree horizontal/13 degree vertical transmission.

OMNI-8 ANTENNA SPECIFICATIONS	
Antenna Gain*	8 dBi
Kit Gain!	6 dBi
Frequency Range	2.4 to 2.5 GHz
Impedance	50 Ohms
Dispersion	360° horizontal/13° vertical
Polarization	Vertical
VSWR	< 1.5:1
Dimensions	17" length; 43 cm
Operating temperature range	-100° to 185° F; -40° to 85° C
Wind load @ 125 mph	5.53 lbs; 2.5 kg
Weight	.45 lbs; 205 g
Mounting	Wall and mast L bracket, 2" mast
Cable Type	LMR 400
Cable Length	20'
Cable Connector	Proprietary SMA male
Cable Attenuation	1.4 dB (ft.)
Cable Impedance	50 Ohms

Figure D-6. OMNI-8 antenna specifications.

Mobile Antenna

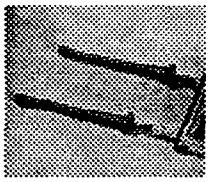


Figure D-7. MMCX-2 mobile antennas.

The MMCX-2, 2 dBi, omnidirectional antennas come in pairs and are used with the BreezeNet PRO.11 SA-PCR PC Card for long range outdoor applications. In addition they can be used in harsh indoor environments such as warehouses, airports, and large storage areas where range can be affected by too much multipath propagation.

The MMCX-2 antenna attaches to the female connector on the SA-PCR PC Card and provides a 360 degree horizontal/60 degree vertical transmission.

MMCX-2 Antenna Specifications	
Antenna Gain*	2 dBi
Kit Gain!	2 dBi
Frequency Range	2.4 to 2.5 GHz
Impedance	50 Ohms
Dispersion	360° horizontal/13° vertical
Polarization	Linear
VSWR	2:1
Dimensions	2.5" length; 63 mm
Operating temperature range	32° to 104° F; 0° to 40° C
Connector	Male MMCX

Figure D-8. MMCX-2 antenna specifications.

Appendix E. Medic-Cam Specifications

Overview

The Medic-Cam system is a small, lightweight, man mobile video and audio coder/decoder used by medical staff to provide expert medical assistance remotely relative to mobile base station (ambulance). The system has been sized to be totally self-contained on an individual soldier's load-bearing vest. The system, which is designed by the Telemedicine Research Lab, is technically integrated using commercial off the shelf technologies by the US Army Research Lab at Adelphi, Maryland. Per unit prototype costs are approximately 18k per unit. The Medic-Cam system consists of a head mounted display with a high-resolution color camera, microphone, controller, antenna, transmitter, receiver and battery (see Figure E-1).

Mobile Unit

The mobile unit is composed of four components including:

- Main System Control

The Main System Control (MSC) unit controls the main power and provides interfaces for all components in the system i.e. HUD, cameras, speaker and transmitter. MSC also provides switching control between different cameras.

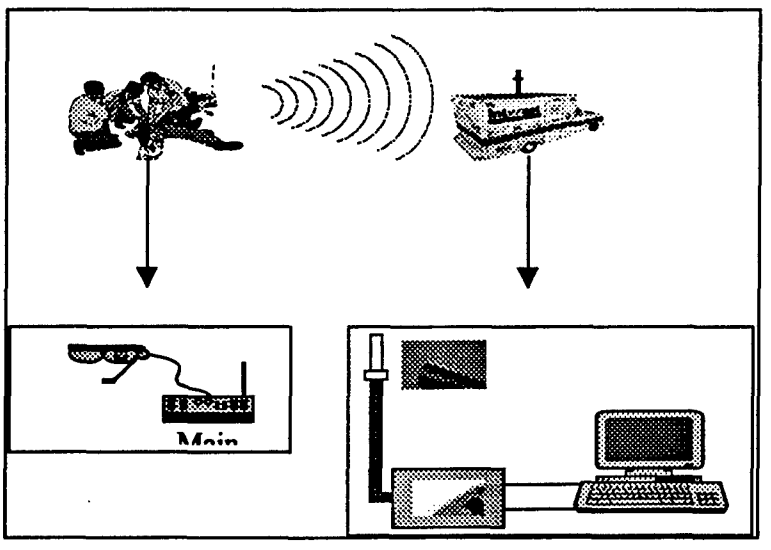


Figure E-1. Medic-Cam System.

- HUD

The HUD eyewear LCD device provides a color image of what is observed by the camera. This visual feedback is required to control the camera's position. The HUD also has brightness control and defaults to medium setting on power up.

- Transmitter

The SR Series of receivers from GMS are advanced technology microwave receivers and operate between 1.7-1.85GHz band with transmission power ¼ or 2 watt selectable. The SR Series demodulates one video and two audio signals from a composite analog RF source. The receiver is controlled by a microprocessor with an easy to use menu system. Channel of operation, video, and audio output levels can be selected via the front panel membrane switch. Operational parameters are displayed on the easy to read liquid crystal display including channel, signal strength, output level, and lock status. Both digital and analog representations of signal strength are incorporated to facilitate path alignment.

The SR Series features an integral Low Noise Amplifier at the RF input with a nominal noise figure of 4dB. The dual conversion receiver employs a SAW filter at each IF stage for enhanced adjacent channel interference rejection. In addition, the receiver is equipped with an internal low pass filter eliminating the need for an external video filter. The SR Series receiver can be factory configured for either NTSC or PAL formats.

Two buffered video output signals are provided at BNC connectors in the SR Series receiver, with balanced audio and composite video available on the rugged 10-pin connector. Low power consumption makes the SR Series ideal for use in remote applications.

- Antenna

A Compatible 1db gain for 1.7-1.85 GHz band.

Base Station

The base station on the receiver side is composed of two components which include:

- Receiver

The SR Series of receivers from GMS are advanced technology microwave receivers and operates at 1.7-1.85GHz bands. The SR Series demodulates one video and two audio signals from a composite analog RF source. The receiver is controlled by a microprocessor with an easy to use menu system. Channel of operation, video, and audio output levels can be selected via the front panel membrane switch. Operational parameters are displayed on the easy-to-read liquid crystal display including channel, signal strength, output level, and lock status. Both digital and analog representations of signal strength are incorporated to facilitate path alignment. The SR Series receiver can also be controlled with the GMS RCB-2000 hand-held controller via a simple RS-232 interface.

The SR Series features an integral Low Noise Amplifier at the RF input with a nominal noise figure of 4 dB. The dual conversion receiver employs a SAW filter at each IF stage for enhanced adjacent channel interference rejection. In addition, the receiver is equipped with an internal low pass filter eliminating the need for an external video filter. The SR Series receiver can be factory configured for either NTSC or PAL format. Two buffered video output signals are provided at BNC connectors in the SR Series receiver, with balanced audio and composite video available on the rugged 10-pin connector. Low power consumption makes the SR Series ideal for use in remote applications.

- Antenna

GMS Omni Antenna with 2db gain for 1.7-1.85 GHz band.

CA/ALO1

Record Retrieved

Document Title : Disaster Relief and Emergency Medical Services Project (DREAMS TM): digital EMS

Next Hit -->

(This is hit 1 of 2.)

AD Number: **ADA386760**
Subject Categories: MEDICINE AND MEDICAL RESEARCH COMPUTER PROGRAMMING AND SOFTWARE
Corporate Author: TEXAS UNIV HEALTH SCIENCE CENTER AT HOUSTON
Title: Disaster Relief and Emergency Medical Services Project (DREAMS TM): digital EMS
Descriptive Note: Final rept. 23 Jan 1999-22 Sep 2000 Should be annual please correct
Personal Authors: Duke, James H., Jr.; Well, James; Salinas, Jose; Tindall, R. D.
Report Date: OCT 2000
Pages: 104 PAGES
Contract Number: DAMD17-98-2-8002
Monitor Acronym: XA
Monitor Series: USAMRMC
Descriptors: *CLINICAL MEDICINE, *MEDICAL SERVICES, *DIGITALIS, REVERSIBLE, CASUALTIES, THERAPY, WOUNDS AND INJURIES, TIMELINESS, CATASTROPHIC CONDITIONS, PHYSICIANS, DEATH, CELLS(BIOLOGY), BLOOD CIRCULATION, DREAMS, INTERVALS.
Identifiers: EMS(EMERGENCY MEDICAL SERVICES), *EMERGENCY MEDICAL SERVICES, *DIGITIAL MEDICINE
Abstract: Physician's virtual presence in support of the first responding care- givers at the scene of the incident will create the opportunity for achieving an accurate initial evaluation of the victim's clinical condition and a timely initiation of appropriate interventions. Any condition that interferes with adequate blood flow will cause an impairment of tissue oxygenation, the results of which are cell injury and, if sufficiently prolonged, cell death. The interval of time between the acute catastrophic events that initiate the decrease in blood flow and the establishment of therapies to reverse this cascade of cell injury is critical. Any measure that shortens the interval between the injury and the institution of appropriate therapy will afford the greatest potential for minimizing cell injury and preventing cell death. The DREAMS: Digital EMS project is designed specifically to address these issues.
Limitation Code: APPROVED FOR PUBLIC RELEASE
Source Code: 396843
Citation Creation Date: 07 MAR 2001

Hard Copy Cost

Based on the document information (104 pages), the cost of this report is $12.00 dollars per copy unless annotated otherwise in the pages field above.